CSDN 技术改变世界 创新驱动中国

Programmer 程序员

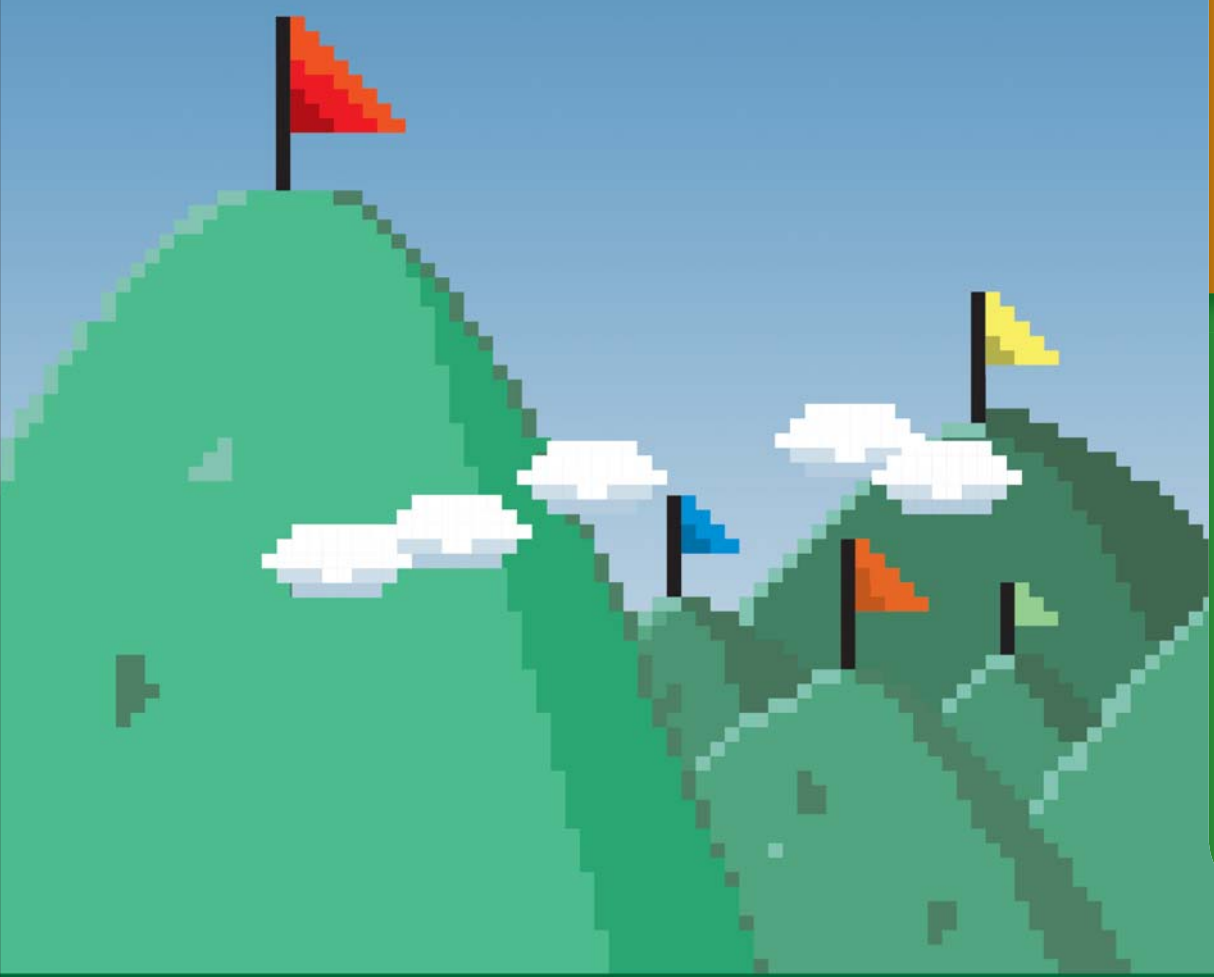
ISSN 1672-3252
9 771672 325111 03>

2011年 3月刊
邮发代号：2-665 定价：15元

2011开放平台之征

六大平台实战分享：淘宝、腾讯、新浪微博、51.com、人人网、SAE

整合产业链是关键 ● 透明是开放平台成功的要旨
如何设计更好的API ● 构建更开放的微博平台
开放平台深度思考 ● 微创业，从开放平台和云计算开始



特别感谢

(排名不分先后，以姓氏首字母为序)

Contributors

Mars Cheng

感谢安天实验室高级产品经理 Mars Cheng，分享 RAS 大会的精彩内容，详见“报道”栏目。

岑文初

感谢淘宝开放平台主架构师岑文初对“架构师接龙”栏目的支持。

方国伟

感谢微软软件架构资深顾问方国伟为本刊撰文，详见“技术”栏目。

高巍

感谢安卓爱普公司创始人高巍就“程序员如何提高职业技能”阐述观点，详见“观点”栏目。

李福松

感谢人人网开放平台高级经理李福松，分享人人网开放平台的相关技术，详见本期“封面报道”栏目。

李严冰

感谢 VMware 中国研发中心总经理李严冰接受本刊记者采访，详见“三八妇女节”特别专题。

李志才

感谢人人网开放平台 API 技术主管李志才，分享人人网开放平台的相关技术，详见本期“封面报道”栏目。

刘龙龙

感谢台湾铭传大学国际学院教授刘龙龙，分享关于云计算和开放平台发展的观点，详见“高端视点”栏目。

刘铁锋

感谢百纳信息技术有限公司 CTO 刘铁锋为本刊撰文，详见“移动专栏”。

罗剑锋

感谢《Boost 程序库完全开发指南》图书作者罗剑锋，分享 C++ 元编程的相关经验，详见“产品报道”栏目。

马博

感谢 UCPM（中国产品经理联盟）发起人之一马博为本刊撰文，详见“TUP 专栏”。

马如悦

感谢百度基础架构部高级工程师马如悦为本刊撰文，详见“论文研读”栏目。

潘正磊

感谢微软 Visual Studio 商业软件部总经理潘正磊接受本刊记者采访，详见“三八妇女节”特别专题。

Todd Papaioannou

感谢 Yahoo 云计算架构副总裁 Todd Papaioannou 接受本刊采访，详见“高端视点”栏目。

孙良

感谢腾讯搜索技术研发中心总经理孙良接受本刊记者采访，详见“对话 CTO”栏目。

滕振宇

感谢 Scrum 教练滕振宇，分享多年来软件工具使用心得，详见“产品推荐”栏目。

王一

感谢篱笆网产品技术部 Web 应用架构师王一为本刊撰文，详见“技术”栏目。

王煜全

感谢 Frost&Sullivan 中国区总裁王煜全分享移动互联领域新观点，详见“评论”栏目。

杨海朝

感谢新浪首席 DBA 杨海朝为本刊撰文，详见“架构师接龙”栏目。

杨卫华

感谢新浪产品事业部技术经理杨卫华，给读者带来开放平台的相关思考，详见本期“封面报道”栏目。

赵宏华

感谢 51.com 事业部副总经理赵宏华，分享 51 开放平台的技术内容，详见本期“封面报道”栏目。

赵劼

感谢盛大创新院研究员赵劼对本刊的大力支持，详见“报道”栏目。

周爱民

感谢支付宝公司业务架构师周爱民为本刊撰文，详见“技术”栏目。

周华林

感谢网票网技术总监周华林接受本刊记者采访，详见“对话 CTO”栏目。

开放的不仅仅是平台

上月诺基亚高调转型中最值得注意的，其实是CEO Stephen Elop在新战略宣布之前发表的备忘录“我们的平台着火了”。无论Elop最后是否能借与微软结盟而力挽狂澜，他在这篇文章中说出的一段非常有价值的话都值得我们认真思考：

设备之争现在已经演变为生态系统的战争。其中，生态系统不仅包括设备的软件和硬件，还包括开发人员、应用、电子商务、广告、搜索、社交应用、地理位置服务、统一通信及其他许多内容。

事实上，这种竞争形式的巨大变化不仅发生在移动设备市场。Apple作为本世纪以来最耀眼的科技巨头，已经通过全新的整合生态系统，先后以iPod+iTunes颠覆了音乐产业，iPhone+App Store颠覆了移动产业。现在，它正在将iPad变成发行平台颠覆新闻出版产业（30%的通道费用让我的同行们无不大惊失色）。

类似的颠覆还发生在更多领域。如果再往远一点看，在云计算、物联网的大趋势下，未来不仅包括计算机（IT软硬件和服务厂商）、通信（运营商、终端和设备制造商）、互联网、媒体内容（广播影视、新闻出版……）等在内的整个信息服务产业将发生全面重组洗牌，众多曾经井水不犯河水的企业要同台竞争，而且全球、全社会原来还没有数字化和接入网络的人、物、流程、信息等等，也将被全部或者部分地迅速纳入和融合到一个拥有持续在线能力、超越地理与时间局限的大平台中，从而对社会的组织形式和人们的生活方式都产生深远影响。

这里说的未来并不遥远，Google之后，社会化、本地化、移动、游戏四大因素剑锋所至，Facebook、Zynga、Twitter、Groupon、Foursquare等等相继崛起，都只用了几年时间，Instagram、Flipboard等新锐获取数百万用户更只在数月甚至数周之间，而它们进化之快、整合能力和发展潜力之大，实在令人心惊。

《华盛顿邮报》一篇文章指出，他们的成功有赖于硅谷独特的通过开放协作来竞争的特殊大环境，而他们也通过贡献力量延续这种传统，使自己继续发展壮大。写博客更多地公开自己的情况和想法、开放API、将一些非关键业务的核心技术开源、建立自己的开发者社区、组织技术活动、建立生态系统……这一系列以前只有大公司才做的事情，现在也成了硅谷创业公司的标准实践。

我们中国的企业呢？如果要在未来这个全球大平台中占据更多的份额，必须尽快向国外的同行学习，抛弃割喉式的过时竞争思维，开放、协作，建设自己的生态系统。

建设繁荣的生态系统并非易事，如何打造坚实的基础平台，如何设计API，如何运营社区，如何分工，如何分利，如何平衡规则和创新……涉及的环节和利益众多，都需要经验和积累。

2010年，我们高兴地看到了许多国内优秀企业已经开始在开放平台上走出实质性的步骤。但仅仅有开放平台的形式是不够的，需要开放的还有我们的胸怀、见识和理念。而且，开放不仅仅是对外的，它还会促使我们自身改变，真正融入到全新的有机生长的生态系统中，成为其中的一部分。

邮件：liujiang@csdn.net
新浪微博：@刘江CE
欢迎大家交流反馈，欢迎投稿、批评、建议和挑错

Contents

Contents

2011年3月刊　总第221期

主管：中国社会科学院

主办：中国社会科学院文献信息中心

出版：《程序员》杂志社

网址：http://www.programmer.com.cn

国际刊号：ISSN 1672-3252

国内刊号：CN11-5038/G2

邮发代号：2-665

广告经营许可证号：京东工商广字0188号

总编：黄长著　Editor-in-chief：Huang Changzhu

社长/常务副总编：张悦校　President：Zhang Yuexiao

副社长：蒋 涛　Vice President：Jiang Tao

编委会：黄长著　张悦校　陈洋彬　蒋涛　曾登高　刘江

Editorial Member：Huang Changzhu　Zhang Yuexiao　Chen Yangbin　Jiang Tao

Zeng Denggao　Liu Jiang

执行主编：孟迎霞　Executive Editor-in-chief：Meng Yingxia

编辑部主任：常政　Director：Chang Zheng

责任编辑：董世晓　高松　郑柯

Editors：Dong Shixiao　Gao Song　Zheng Ke

特邀编辑：方梁　高昂　赵健平　吕娜　卢鹃翔

Contributing Editors：Fang Liang　Gao Aang　Zhao Jianping

Lv Na　Lu Dongxiang

美术设计：纪明超　Art Designer：Ji Mingchao

美术编辑：林象海　Art Editor：Lin Xianghai

流程编辑：陈秋歌　Coordinator：Chen Qiuge

Tel：010-64351458

Email：editor@csdn.net

发行部　Distribution Dept.010-64351431

Email：sales@csdn.net

广告总代理：北京创新乐知广告有限公司

Sole Advertising Agency：Bejing CSDN Co.,Ltd

Tel：010-64376055

Email：ad@csdn.net

Marketing Dept：010-51661202（ext 149）

Email：market@csdn.net

读者服务部

Readers service Dept.

网上订购：http://dingyue.programmer.com.cn/

读者信箱：reader@csdn.net

地址：北京市朝阳区广顺北大街33号院1号楼福码大厦B座12层

Address: B-12th Floor Fairmont Tower NO.33 Guangshun North street,

Chaoyang District,Beijing

邮政编码：100102

电话：010-64351436

传真：010-64348545

法律顾问：北京中润律师事务所 王杰

Law Consultant：Beijng Hengsheng Lawyer Firm

印刷：北京盛通印刷股份有限公司

Print: Beijing Shengtong Printing Co., Ltd.

出版日期：每月1日

Publication Date：the first day per month

零售价：RMB 15.00元　新台币 390元　HK $ 35.00（港、澳）

US $ 9.00（海外）

Retail Price：RMB 15, NT$390, HK $ 35.00,US $ 9.00

EDSAC计算机之父

——Maurice Wilkes

■ 文 / 苏椰

1951年5月15日，在英国，BBC频道播出了三个人的演讲，其中有两位大名鼎鼎：一位是计算机科学之父艾伦·图灵，另一位是图灵的导师、剑桥数学家纽曼。那么第三个人是谁呢？谁有资格跻身如此阵容呢？

他叫Maurice Wilkes。

Wilkes，1913年出生于英国，1931年进入剑桥大学，后又进入卡文迪什实验室。1936年获得物理学博士学位，论文题目是《关于甚长无线电波在电离层中的传播特性》。二战爆发后，Wilkes为英国军方研制雷达设备，辗转于敦刻尔克、剑桥、马尔文等地，参与过10cm雷达和OBOE的研制。OBOE是一套轰炸机导航系统，可以使飞行员不需要地图和目视，只需要按照地面发来的指令飞行，即可准确到达轰炸目标。地面上有一个“猫站”和一个“鼠站”，猫站会通过信号，将飞行员引导到一个大圆弧上（轰炸目标就在这条弧上），如果飞机在弧内，猫站就发射“点”信号，如果在弧外则发射“划”，飞行员就根据这些反馈来修正方向，保持位于弧线上。当飞机位于目标上空时，鼠站就发出信号，飞行员就投弹，然后再用同样方式，由猫站引导回基地。这套系统大受盟军飞行员的欢迎。

1945年，战争结束了。Wilkes回到剑桥，主导了英国的第三个电子计算机工程。读者可能会问，前两个是什么？第一个是“巨人机”，战时诞生于布莱切利庄园，开发者是艾伦·图灵。第二个是ACE（自动计算引擎），国家物理实验室正在研制中，此时的主导者也是艾伦·图灵。Wilkes扛起了第三个，一个名为EDSAC（电子延迟存储自动计算机）的计算机工程。起初他联系了国家物理实验室，希望得到一些资料，但当时ACE还没有正式起步，而美国冯·诺伊曼的EDVAC还是秘密，所以Wilkes只得到了一些零星的见闻，以及被安排在1946年到宾夕法尼亚参加ENIAC团队主办的一系列讲座。1946年11月，Wilkes希望参与ACE项目，他于11月27日访问了国家物理实验室，并于12月2日给图灵写信谈了许多关于ACE的设计想法，并附上了他得到的一些关于EDVAC的资料。但这封信的问题在于，Wilkes并不知道图灵已经做了7个版本的设计工作，而且EDVAC的一些特性正是受到图灵的启发。更糟糕的是，Wilkes的想法，很多是与图灵的原则相反的。图灵的设计哲学是尽可能地简化硬件，把其他的事情都留给程序。图灵在回信中表达了这些想法，所以合作并非真正展开。1947年，他们的联系彻底中断了，而此时由于ACE项目出现了一些人事上的问题，谁也不肯让步，最终图灵一气之下甩手不干了。Wilkes的EDSAC项目仍在继续，他转而与EDVAC的项目人员取得了广泛的联系，他们经常一起切磋讨论。EDSAC采用水银延迟线作为存储器，容量是512×18位，加法时间1.5毫秒，乘法时间4毫秒。威尔克斯还引入了变址、缓存、宏指令、微程序、子例程等重要概念，还设计了一个子例程库，这些都对后来的计算机设计产生了极其深远的影响（有些资料认为这些都是Wilkes发明的，但笔者认为不然，很难考证最先提出者，但至少图灵在很早之前就有微指令和子程序的想法）。在工程实施中，由于资金缺乏，项目一度岌岌可危。最后关头，Wilkes说服了一家面包公司来投资，终于绝处逢生。1949年5月，EDSAC首次运行，这是世界上第一台冯·诺伊曼结构的计算机。戏剧的是，之前一直作为“老师”的EDVAC项目，反而直到1952年才完成。

1967年，ACM授予威尔克斯图灵奖，以表彰其在存储程序式计算机和子程序等重要概念上的杰出贡献。1980年，Wilkes从剑桥退休，出任DEC公司的顾问和MIT兼职教授，1986年回到英国，担任Olivetti公司的顾问。2000年，Wilkes被授予爵位。两年后他搬回剑桥，担任剑桥大学荣誉教授。2010年11月29日，一生声名卓著的Wilkes离世。

开源云计算：Yahoo的创新驱动力

Todd Papaioannou
Yahoo云计算架构副总裁

经常有人问，Yahoo准备转向云吗？我们的回答是，不，我们已经是云了。Yahoo不会提供Amazon或者Google那样的公共云平台。但是，我们早就开始向数以亿计的用户提供个人云服务了：邮箱、照片、金融服务等等。我们称之为个人云。

更重要的是，当业界目前更多地将云计算视为降低成本、节约能源手段的时候（这些当然也非常重要），在Yahoo，云计算已经成为一种关键性的创新驱动力。

作为全球最大的互联网公司之一，Yahoo正面临着巨大的技术挑战。公司自身拥有庞大的网络资产，超过9千万网页，6亿用户（仅Yahoo邮件就有超过3亿的用户），成百个关注点和背景各异的产品和服务，每天要通过分析一千亿以上各种各样的事件：登录、提醒、广告点击、文章点击、论坛发贴、上传图片、打标签、购物车……每天的流量数据以PB计算，存储数据量更是以数百PB的速度增加……

怎样才能在如此大规模的平台上，快速从海量数据中提取有价值的信息，将最受欢迎的内容提供给对其最感兴趣的用户，满足各种各样个性化的使用模式？怎样在这种规模的平台上，将停运时间降至最低（在Yahoo，即使是短时的停运，损失都将高达数百万美元），满足用户不断变化的需求，提供更好的用户体验？怎样优化Yahoo的现有产品与服务，提升广告商的满意度，从而提高公司的收益？

应对这些挑战只能依靠创新，而创新又有赖于云计算基础设施的支持。与其他公司不同的是，Yahoo在云计算方面采取了全面开源的战略。众所周知，Yahoo是开源云计算技术平台Hadoop的诞生地和主要支持力量。在过去五年多时间里，Yahoo在Hadoop以及Pig、ZooKeeper、Hive、Howl、HBase和Oozie等相关开源项目中投入了大约300人年，累计数千万美元，将Hadoop从一个有趣的原型发展为坚实的可扩展框架，产生了丰硕的成果。

Hadoop也已经成为Yahoo基础设施和许多重要业务流程（搜索、广告、反垃圾邮件、个性化等等）的核心组件。Hadoop在Yahoo内部已经广泛应用于多个生产环境，涉及全球多个数据中心，超过4万台服务器（内含30万以上的CPU核心），20多个集群。其中最大的集群包括4千台服务器，也是世界上规模最大的Hadoop集群。目前Hadoop支持着公司内部1000多个科研团队用户，每天超过20万个作业，每秒几万次请求。甚至可以说，在Yahoo各个网站上每一次点击背后都有Hadoop的功劳。Hadoop使Yahoo更多研发人员可以在更高的抽象层次工作，大大缩短了产品开发周期，显著减少了人力和基础设施成本。

未来，Yahoo还将对Hadoop等云计算基础设施研发和社区支持继续投入。而且，我们正计划通过Hadoop和其他开源项目，将Yahoo内部所有的底层云计算基础架构逐步地全部开源。为什么我们这样大力支持开源？原因很简单，我们不认为这些云计算基础技术是什么差异化竞争优势，而且Yahoo已经从Hadoop活跃的开源社区中获益匪浅。

从Hadoop的成功故事中，我们可以总结以下几点开源的优势：

通过开源，Hadoop已经从一个内部技术成长为优秀而稳定的工业标准，从而避免了一般企业内专有技术经常遇到的被外部新标准逼向死胡同的问题。

通过开源，Hadoop社区在Yahoo之外出现了更多活跃用户，他们的贡献产生了许多对Yahoo也很重要的技术，比如HBase和Hive，最终节约了公司的成本。

通过开源，Yahoo公司能够从社区更容易地聘请到优秀的训练有素的人才，而且与许多伙伴的合作也更加顺畅。

更重要的是，通过开源，我们既能够以最经济高效地方式进行研发，实现自身的业务目标，又能够欣喜地看到自己的工作被成千上万的人用于远超出预期的各行各业，最终改变了世界，我们为此而深感自豪。

云计算与开放平台

刘龙龙
台湾铭传大学国际学院教授

美国国家标准与技术局（NIST）所草拟的一份说明，可能是对云计算基本概念解释的最佳范例。其实它的主要内容只有很少的一页半，但是里面的许多专有名词还是需要进一步去了解才能体会。事实上，依照目前云计算的进展，它是不需要特别去定义或规范的，只要代表着一个通用的示意名词就行；一步步来，不用急着把自己绑住。回想50年代的计算器以及程序员工作都是极少数人才能接触到的领域，可是从80年代开始，个人计算机出现、微软窗口操作系统称霸、亚洲地区大量生产制造计算机等等因素促成IT应用的普及率迅速提升。这30年来，从个人计算机→网络→服务→云计算的进步模式，是一种持续发展的技术与应用。这些名词强调的是概念或模式，而不是严谨的定义（学校里学生考试时另当别论）。当然，现在一般人对于个人计算机与网络的认识与经验是比较多的，而相对之下对于服务与云计算的概念就比较陌生。

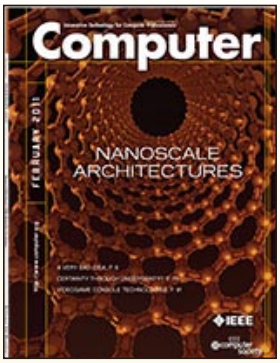
如果从参与者角色的扮演来看云计算，也许会更清楚一些。假设角色是简单地分成：一般的用户（或消费者）、程序员、想要赚钱的企业或老板三种，而每个人都可以尝试扮演这些角色。从使用者来看，他们不会对技术细节有太多的兴趣，可是会想要知道自己从云里面到底能得到什么样的服务，然后再算算价钱是不是划得来。比如说，一个人家里的个人计算机已经是一两万块钱人民币的高档货，上头什么软件都装得差不多了，而且网络带宽也很充裕，参加社交网络活动也很积极；那他可能想不出云计算会对他有什么特别的帮助（因为他不需要云里面提供的服务就很方便）。因此，云计算所提供的服务一定要能带给使用者比他用他现有的基本设备与网络带宽所能获得的效益更多、更好（或他原来根本做不到的事情），才会产生吸引力。这大致也说明了云计算所标榜的On Demand与非云的On Premises之间差异的基本概念。

从程序员来看，他们很想立刻就知道在技术上云是怎么样建置起来的。但是，在了解到云计算都是大手笔（包括数据中心设备以及大量复杂的软件）的老板们才能玩得起的项目之后，他们会有两个选择，一个是到云计算公司去当伙计，另一个就是搭云计算的开放平台顺风车。开放平台这个名词又是一个概念(虽然目前中文维基百科里还没有它的内容说明)，它是由开放及平台两个名词连起来而成的，所以理论上它应该充分代表开放的精神（包括开放授权、开放架构、开放文件、开放源码等等）才是。平台比较单纯，只是它也已经从硬件、软件衍生到网络、服务，以及云计算的范围内。程序员现在认知的开放平台包括如linux或Android及ADK套件以及Google App Engine等等。利用这些开放平台，不管是不是有经验的程序员都可以在上面轻易地开发各种应用软件，同时在Marketplace上直接贩卖抽成。虽然有关于开放平台应用的伦理与道德、商业逻辑、创业等讨论是另一回事，但是开放平台是由前辈程序员开发出来给后辈程序员来使用以节省开发时间、增进软件质量的传统精神，是不变的。

从想要赚钱的企业或老板来看，云计算的市场及业务机会似乎就在眼前，但是问题在于自己如何才能分到一杯羹。比如说，Google与Amazon对既有的网民所提供的服务水平已经是遥遥领先，IBM与Microsoft则是针对他们既有的企业客户群继续努力提供服务支持、步步为营以避免客户流失，OpenStack的成员们也期望在开放平台被普遍认可、支持的概念基础上另创机会。台面上下，大家都在竞争，希望能先冒出头来。其实，在软件技术上要追上Google或Amazon并不是不可能，比较难的应该是累积的人气与商誉（究竟它们已经营运十几年了，何况还一直在创造新的业务模式）。IBM与Microsoft则是对企业客户有承诺，虽然使用他们的产品及服务有被套牢的顾虑，但任何新技术出现他们一定会提出相应的产品来维持客户的业务竞争力。另外则是对于旧东西提供的兼容性考虑，这也是他们可以多年来继续获利、生存的一个重要因素。至于新加入的OpenStack成员，拭目以待无妨，他们必须抓住使用者或客户的心并提供所需才能赢过领先者。

IEEE Computer
2011.02

纳米架构



本期《IEEE Computer》中，有4篇文章主要介绍了为应对后硅器件时代的计算带来的挑战，而提出的几种新型架构。另外几篇文章对网站验证码的安全性及软件产品线工程中的形式化方法进行了讨论。

在《从突触到电路：使用忆阻式存储器搭建电脑》一文中，作者提出了一种新的架构，来模拟人脑的网络结构与算法。为了实现这种规模的网络结构，需要和生物突触一样超级小的电路。本文描述了如何忆阻式纳米器件与PIM架构相结合，构成纳米级别的仿人脑计算机。

《使用新型浮动闸设备进行计算》一文的作者介绍了在低能耗FPGA和模数转换器等非传统的应用中，单浮动闸和双浮动闸器件的设计、工作方式以及对架构的影响；并且提出了一种统一的非易失性/易失性存储设备。

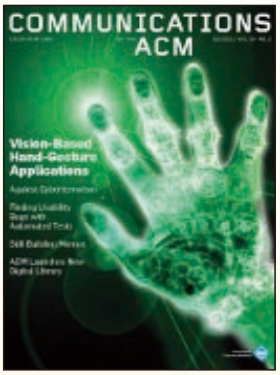
《水晶和雪花：用纳米线交叉结构构造计算机》一文展示了如何完全抛开硅构成的微电路构造一台计算机。作者将纳米线器件排成类似水晶的规则晶格结构，开发出了一种纳米计算机架构。

《迈向纳米器件构造的未来系统：应对可靠性挑战》一文指出，纳米级电子器件被认为可以在器件的密度、能耗和性能等方面带来巨大的改善，但是它们能否成为现实，取决于我们是否能应对好可靠性的挑战。同时，纳米器件展现出的新特性也要求我们重新考虑可靠性策略。

验证码是商业网站中采用的标准防御措施，但却被简单但是新颖的攻击方式攻破。《从安全工程的角度看验证码的健壮性》一文中，从安全工程的专业角度给出了对验证码设计的建议，可以大大提高验证码的健壮性。

Communications of ACM
2011.02

技术前沿与思考



本期的《Communications of ACM》介绍了人机界面、数据挖掘、可用性测试方面的新技术；并对虚拟现实、网络恐怖主义的威胁进行了探讨。

人机接口向着越来越自然、直观的方式发展，越来越接近人与人的沟通方式。身体姿态和手势就是人机互动的一种自然方式，但是首先系统要能辨认出它“看”到的事物，《基于视觉的手势控制应用》一文中，介绍了手势控制人机界面，以及在满足不同应用时面临的挑战，包括医疗辅助系统、危机管理和灾害救援、娱乐，以及人-机器人互动等应用。

互联网是充满了共享文档的宝库，但其中也包含着很多结构化数据。《互联网上的结构化数据》一文讨论了互联网中结构化数据的特性，以及对其进行管理会遇到的挑战。还有谷歌的“网络表格”和深层网络爬虫是如何发掘并提取出之前难以获取的数据资源，并展示给终端用户。

美国国家工程学院将虚拟现实列为21世纪待解决的14个工程问题之一。《关于虚拟现实的10个科学难题》中，作者探讨了10个与虚拟现实技术相关的挑战。

网络恐怖主义是在当代媒体中反复出现的一个字眼。但是《应对网络恐怖主义》一文的作者则认为，时至今日，还没有出现真正的网络恐怖主义，而且由于网络恐怖主义的成本高于其引起社会恐慌的效果，所以在近期，未有网络恐怖主义之虞。

如今，网络战、网络攻击、网络犯罪、网络恐怖主义的威胁空前，而这些行为大多通过恶意软件实施。《忽视恶意软件教育的危害》一文再次探讨了培训反恶意软件专业人才的必要性。

MSDN
2011.02

动态编程

本期的《MSDN》讨论了动态编程问题，包括动态数据以及C# 4中出现的动态关键字。另外还有几篇文章分别介绍了Windows Phone 7编程技术、工作流服务的安全以及SynchronizationContext类。

在将其他来源的数据合并入我们的应用程序使用的数据库时，可能会需要进行匹配和重复删除步骤，已得到最终可用的数据库。《动态数据：使用F#进行模式匹配数据库记录》一文中，展示了4种不同的匹配算法，并用F#编程实现。

动态关键字为C# 4带来了激动人心的新功能。《理解C# 4中的动态关键字》一文介绍了它的工作原理，以及为何它能简化大部分的编程工作。

Windows Phone 7拥有一个巨大且不断增长的资源系统，可供开发者使用。《Windows Phone 7开发工具和资源》一文总结了其中的几个主要资源。包括微软的App Hub、Channel 9上的视频教程、XNA的布局框架XPF等等。

在为智能手机编写程序时，将用户界面进行精简，只留下必要部分，是很重要的。《Windows Phone 7上的声音录取》一文中，作者带我们看到了一款应用可以做到怎样精简的程度，同时保持其可用性。而且在这个过程中，会介绍Windows Phone 7上的录音API。

在《保护WF 4工作流服务》一文中，作者讨论了针对不同工作流宿主，可以采用的不同安全措施。其中包括了工作流安全包，它的一系列活动可以用来为工作流解决方案提供端到端安全。

SynchronizationContext类是.NET中多线程组件能够正确运行的中心一环，但是常常被忽视或者误解。《关于SynchronizationContext类的一切》介绍了此种类的必要性、概念、实现方法、注意事项，以及其库支持。

观点

@dlee_cn

Fielding本人也不会想到REST架构思想的影响力会有这么大，现在REST已经极大地影响到了Web服务（云计算，间接）、SOA、Semantic Web、Mashup、企业级Portal等直接相关的技术领域。

@soulhacker

制作精良的APP如Twitter和TweetList，无论怎么刷新，已有推位置是一个像素也不会动的，只在上面“长出”新推；差一点的TwitBird刷新完列表会移动一点；而所有新浪微博客户端都是直接把你扔到最顶上，给它提建议得到的反馈是“大多数用户习惯往下翻”。

@iamxhu

在看《Coders At Work》一书，作者职业生涯很传奇。早年记者，20世纪90年代初搞Perl，之后Java革命，加入WebLogic，之后又搞Lisp，花两年时间搞出《Practical Common Lisp》一书。

@jamespearce

“移动互联的新语言”，显然不是JavaScirpt， http://bit.ly/g6oq71。

@huandu

开源轻量级HTML5游戏引擎gin简介：HTML5 Canvas让程序员自由地绘制想要的图形和动画，使得纯粹基于HTML/JavaScript/CSS的游戏铺平了道路，http://goo.gl/fb/QVvQf。

@jjgod

EPUB3的草案（http://t.co/XwZngQ0）目前看起来太庞大了，不切实际了一点，这样一来开发一个EPUB3阅读器的难度比开发一个浏览器还要大，最后的结果很可能就是实现分裂和垃圾阅读器泛滥。

资源

@flycondor

优雅的Bitcask， http://is.gd/jxOwP。

@huihoo

开源防火墙pfSense（基于FreeBSD）、Shorewall（基于Linux,让iptables变得简单）、Vyatta（基于Linux,类似Juniper Junos Cisco IOS)，http://wiki.huihoo.com/wiki/Firewall。

@smallfish_xy

MIT Python教程，http://is.gd/aHPnqg。

@bluedavy

推荐一个同事写的关于Hot Code Replace的PPT，写得很好，同时也给大家介绍了一个靠谱的实现Hot Code Replace的方法，http://goo.gl/dtb9i。

@laogao

N组flashcard（适用于复习知识点），包括有Scala、Emacs、Git、Haskell、Clojure、F#等多个主题，http://yuvimasory.com/flashcards.html。

@dingdeng

Internet-history列表邮件不多，但是有很多骨灰级人物在里面活动筋骨。比方说，Ken Olsen一挂，Vint Cerf就又跳出来了，http://j.mp/f4s1et。

@cntom

企业升级，从1.0到2.0，http://bit.ly/fTcx9j。

文章

@xyb

http://t.co/NXmVS2y，持续集成之“测试三角形与分段构建策略原则”。重新调整单元测试、集成测试、验收测试的比例，有助于持续集成的提速。

@qichangxing

其实问答网站，当年奇虎就做过了，没做起来，其中原因值得深思。而百度知道做起来了，这有必然性，因为它是搜索（提问）入口。http://g01.cc/post/260 ，这里有则文章说到初始用户的重与轻的问题，可以参考阅读。

@lazycai

李洋新作：《df、du、fdisk：Linux磁盘管理三板斧的使用心得》。推荐对相关命令有理论基础但还没有多少实际操作经验的管理员们阅读，http://os.51cto.com/art/201012/240726.htm。

@imtomgilb

我的论文：效率即是有效发布，http://bit.ly/hdFflt。

（网址链接请访问杂志官网，www.programmer.com.cn）

CSDN最受关注文章 TOP10

（2011年1月26日~2月26日）

1.	腾讯有多大：10条让你尖叫的数据
2.	IE浏览器走势：IE9涨势喜人IE6往下滑
3.	谷歌成下一个微软式企业10大理由
4.	周鸿祎：“站在对方的鞋子上”思考
5.	多国宣布采用中国4G标准
6.	分析称诺基亚选择与微软合作是一个错误
7.	雅虎继续人才流失，一年半出走15名高管
8.	HTTPS的七个误解
9.	Web2.0版问答网站上线，融入社交功能
10.	Facebook开发新型图片浏览器

JavaEye最受关注文章TOP10

（2011年1月26日~2月26日）

1.	10个优秀网站助你学习Web设计
2.	分布式设计很简单——guzz分布式切表功能正式发布
3.	MySQL前雇员成立新公司SkySQL与甲骨文竞争
4.	Android 2.4将在4月发布，2.3被略过
5.	苹果开始生产iPad 2代，高价Xoom难以挑战iPad
6.	苹果开发iPhone mini，挑战Android
7.	微软发布IE9 RC版
8.	诺基亚与微软合作：将采用Windows Phone
9.	Firefox 4 Beta 11 发布
10.	Oracle提名SouJava成为JCP执委以取代Apache

且说 UML 2.4

主持人：
潘加宇，UMLChina首席专家，潜心研究和实践UML／UP相关技术的应用。

软件工程与管理

2011年1月26日，Steve Cook在博客上发布消息：UML 2.4的所有技术环节已经完成，目前只需等待进入OMG的投票流程，然后将其发布为最新的UML规约。

Steve Cook是微软在OMG的代表。重新加入OMG以后，微软一直积极地参与UML新版本的制定工作。

自2005年7月OMG发布UML 2.0以来，UML版本变更的步子迈得并不大，上一个版本是2010年5月发布的UML 2.3。这是正常的，毕竟建模语言的抽象级别更高，所以相对而言实现语言如C#、Java等版本变化更加频繁。

根据Steve Cook的描述，UML 2.4的主要注意力放在如何使UML规约更加严密上。UML 2.4规约本身就是一个用UML 2.4描述，并序列化为XMI 2.4文件的模型，可以导入到各种兼容UML 2.4的工具中。

与此同时，UML 2.5的制定工作也已经开始。UML 2.5的主要任务是以使用者和工具厂商的视角做进一步的改善。

UML的诞生对软件工程的影响是巨大的。翻开近十年出版的软件开发书籍，其中要是提到建模，使用的表示法大都是UML。建模工具的数量和种类出现了爆炸性的增长，形成了一个产业。UML也被ISO吸纳为标准：ISO/IEC 19501和ISO/IEC 19505。

不过，不少开发人员并不喜欢用UML，更喜欢在白板上画个自造的草图，似流程图非流程图，似类图非类图，然后说“来，我给大家讲讲！”这样的做法有一个巨大的“优点”：因为怎么画都是对的，关于这个草图的解释权归“我”所有，同事也不好批评我，项目主要依赖于“我”头脑中的隐式知识。这一点，在有一定资历、但又不对项目的成败承担首要责任的“高手”身上表现得更明显。

但是，这样的做法更像是想通过形式上的丑陋来遮掩内容上的丑陋。动乱年代，数学家在牛棚中用马粪纸做数学推导，不代表就可以因为演算工具简陋就可以允许自己胡乱使用符号和概念；过去的作家没有电脑，不意味着作家可以随意写错别字犯语法错误。

开发人员故意选择简陋的形式为简陋的内容开脱，就如同作家故意选择不好的纸来掩盖自己文字功力不足的事实，并不是好现象。

就像数学符号背后隐含着数学的基本共识，五线谱背后隐含着基本乐理一样，UML背后隐含的是对于软件建模的一些基本共识。

这些符号幼儿园的小朋友都会画，但背后的共识需要一定的训练和学习才能掌握。

在基本共识上沟通，效率会高得多，无效的低水平争论也会少得多，背后的脓包也会强制性露出来。

总之，开发人员如果习惯于画“草图”，用“模块”、“特性”等词汇含糊不清地表达思想，在严谨建模思维的追问下，往往会千疮百孔，暴露许多之前没有想到的问题。

本月热点

事件

OOP 2011大会

2011年1月24—28日，OOP 2011大会在德国慕尼黑举行，POSA序列书籍作者Frank Buschmann 在会上做题为Software Architecture Paradigms and Styles的演讲。

资源

2010年5月发布的UML 2.3规约，包括PDF和XML文件。

http://www.omg.org/spec/UML/2.3/。

在线系列音频：七分钟学会UML

http://www.podfeed.net/podcast/UML+in+7+Minutes/6518。

一套帮助学习UML的在线自测题

http://www.peoplewarecn.com/UMLQuizLevel1Part1.swf。

主持人：
高昂，中国标准化研究院助理研究员，从事信息技术标准化研究工作。关注开源社区，也是OSGeo中国和InfoQ中文站成员。

JavaScript 代码生成器——CoffeeScript

编程

本月热点

●新品发布

Perl语言发布5.12.3版本

Perl开发者新近发布了Perl 5.12.3版本，这个新版本用了4个月的开发时间，涉及到2500多行的代码改动。

Tangram框架发布1.3.4版本

Tangram框架是百度贡献给开源社区的JavaScript框架，在发布前经过了百度专业QA的测试和百度各产品线巨大流量的考验，所以具有一定的质量和成熟度。Tangram功能丰富，使用简单，容易扩展，能帮助开发者提高开发效率和代码的质量。Tangram以其模块化的架构设计，多浏览器支持和方便定制与扩展的特性，在1.3.4版本发布后引起不少开发者的关注。

●资源

JCoffeeScript

CoffeeScript的Java类库，使CoffeeScript代码在Rhino环境中编译运行。

项目地址：https://github.com/yeungda/jcoffeescript。

Coco

Coco是CoffeeScript的方言，Coco的开发目的是在牺牲CoffeeScript代码可读性的同时，让CoffeeScript语法变得更为高效和实用，以便于应用的快速开发。

项目地址：http://github.com/satyr/coco。

Vim CoffeeScript

该项目为使用Vim文本编辑器的开发者提供CoffeeScript语法高亮显示，同时为开发者提供CoffeeScript代码缩进支持。

项目地址：http://github.com/satyr/coco。

在Web前端开发领域，HTML、CSS和JavaScript的组合以其实用性和通用性而被广泛采用。在越来越多网站开始使用HTML5技术来展现Web应用的同时，各式各样功能强大的JavaScript框架也成为开发者不可或缺的重要工具。JavaScript框架利用其丰富的API和透明的跨浏览器支持来帮助开发者优化编码的过程。此外，为了改进JavaScript语言本身已经固有的缺点，目前已有开发者开始着手创建一种能够通过编译生成JavaScript代码的新语言。

CoffeeScript就是这样一种使用Ruby编写的编程语言，旨在通过简单的编码方式让开发者在Web应用中生成并使用JavaScript代码。CoffeeScript融合了Ruby的简洁和JavaScript的灵活，让开发者通过简单易读的语法来撰写逻辑规则。CoffeeScript编译器负责将代码逐行解释为等效的JavaScript程序，并保证生成的JavaScript代码与源代码在逻辑结构上一致。

较之JavaScript语法，CoffeeScript在语法设计上更为严谨。为了让代码看起来更加简洁精致，CoffeeScript使用空格缩进而非大括号来进行代码段分隔。开发者可以在CoffeeScript编程中调用任何已有的JavaScript类库，并能够与类库实现无缝集成。经编译生成的JavaScript代码，具备良好的版式和可读性，相比较开发者手写的JavaScript代码具备同等甚至更优的执行效率。

CoffeeScript翻译器使用CoffeeScript语言本身编写，能够在编译的时候格式化JavaScript代码，这一点在CoffeeScript官方站点上也有所体现。站点为开发者提供了交互式的CoffeeScript在线编译环境，开发者在一侧敲入CoffeeScript代码的同时，代码被实时解释为对应的JavaScript程序，并可随时在浏览器上运行验证。

CoffeeScript自2009年12月诞生以来，经历了一年多开发完善，前不久刚发布了1.0版本。在新版本中，CoffeeScript改进了循环体结构，并提供了简便的闭包实现。

为了方便Java开发者使用，CoffeeScript还提供一个被称作JCoffeeScript的Java类库，让CoffeeScript代码能够在Rhino环境中编译。Rhino项目由Mozilla基金会赞助，完全使用Java语言开发了JavaScript引擎，以便嵌入在Java应用中为用户提供脚本支持。

目前已有不少Web开发项目开始使用CoffeeScript生成的JavaScript来构建站点。比如提供各种新技术资讯的arstechnica.com站点，在其推出的iPad应用中，就是使用HTML和CoffeeScript为读者提供前端展示。

37Signals发布的iPad白板应用Chalk也是使用CoffeeScript编写，并且能够借助Cache Manifest实现应用的离线运行。运行在浏览器端的坦克大战游戏Orona同样借助CoffeeScript实现。

此外，站点thelincolnshirepoacher.com还使用CoffeeScript和JavaScript的矢量图形库工具Raphaël来演示如何随机生成各式各样有趣的图案。

企业应用开发中的平板电脑

主持人：
邢波涛，现任北京新软孚信息技术有限公司技术负责人。关注SaaS管理软件和B2B、B2C电子商务的融合。新一代电子商务和SaaS管理软件积极的实践者。

企业应用与行业软件

最近看到一则新闻：平板电脑进攻商务领域，传统PC何去何从？在企业应用开发领域，我们几乎所有的软件，无论是应用程序，还是UI本身，都是围绕PC机来开发的。比如，网页最常见的分辨率都是按照1024×768来设计的。1024×768则是PC机以及笔记本的标配分辨率，而最热门的平板电脑除了苹果的iPad分辨率达到了1024×768之外，其余Android系统的平板电脑，一般是1024×600或者更低。当然，也不排斥随着时间的推移，平板电脑的标配也都达到了1024×768。比如，2011年最值得期待的摩托罗拉XOOM，屏幕分辨率达到了1280×800像素。如果平板电脑真的取代了PC，那么它将对企业开发领域带来深远的影响。

首先，在服务器端，对操作系统来讲，无论是Unix/Linux还是Windows Server，都不会发生根本性的改变，毕竟，平板电脑只是应用在客户端。但是在客户端，Apple的iOS和Google开源的Android将占据主导地位，而Microsoft的Windows Phone则会处于小众的地位。无论是iOS还是Android，都是支持HTML5规范的，所以，在企业应用领域，平板电脑的影响，说到底，还是浏览器的竞争。即基于iOS的浏览器和基于Android的浏览器，是否支持自己所开发的企业应用程序。比如：基于Flash技术的Flex，在企业开发领域应用越来越广泛了，但在iOS上就有问题。不过，Apple的平板电脑也不是为企业应用使用的，我们在平时录入数据和使用Office等办公软件的时候，不可能使用触摸屏滑来滑去，来完成各类单据的录入。反而是有键盘的摩托罗拉 XOOM，很可能会满足传统PC的各类操作。所以，基于iOS的企业应用程序，可能仅限于某些特定领域了。

其次，对传统的客户端应用程序来讲，Apple的平板电脑适合输入很少，输出很多的消费领域。而Android仅支持少数几类开发语言，Visual C++、C#、Delphi、Visual Basic这类Windows专用语言，可能是最大的牺牲品。所谓三十年河东、三十年河西。所以，现在腾讯的QQ支持iPad和iPhone，还是很明智的。

最后，对于专注于企业开发领域的软件公司来讲，我建议彻底抛弃C/S领域的新研发，即使做客户端，也不要使用Windows的专用语言来开发，否则，根本无法移植到最新的平板电脑上。而对于服务器端产品的研发，虽然Windows还会继续占据很大的市场份额，基于Windows的.NET还会继续有广阔的用武之地，但在浏览器这端，就不建议使用任何Windows特有的技术了。比较安全的是使用标准HTML和JavaScript语言。

虽然Intel称“平板电脑取代PC”的看法言过其实，但在商业领域，其冲击也是显而易见的。任何技术都不会永生。依附于PC上面的企业应用，也到了变革的时机。

本月热点

新品发布

Apache Camel 2.6.0 发布

Apache Camel 是一个非常强大的基于规则的路由以及媒介引擎，该引擎提供了一个基于POJO的企业应用模式（Enterprise Integration Patterns）的实现。Apache Camel 采用URI来描述各种组件，这样你可以很方便地与各种传输或者消息模块进行交互，其中包含的模块是采用可插拔的方式进行工作的。

事件

顺网科技斥资4.8亿元收购5家公司

1月30日，顺网科技对外发布公告称，将斥资4.8亿元收购5家上海软件、广告类公司。这5家公司分别是：上海新浩艺软件有限公司、上海凌克翡尔广告有限公司、上海派博软件有限公司、上海信御计算机科技有限公司以及上海翊广信息技术有限公司。除了上海凌克翡尔广告有限公司是广告公司外，其余看名字都是软件类公司。顺网科技是杭州的一家做网吧客户端软件发家的公司，在深圳创业板上市。

会议

第九届软交会在京举办

1月24日下午，第九届软交会在京举办媒体见面会。鸡肋一样的具有强烈中国特色的软交会，啥时候能像CES2011国际消费者电子展一样，在软了很多年之后，突然给力一把呢？而不是什么热，就跟风讨论什么，并说自己完全具备自主知识产权呢？

主持人：
姚磊，Microsoft Dynamics CRM MCP，多家企业信息化商业解决方案项目经验。熟悉IT规划与需求工程与项目管理。

物联网、云计算与智能终端应用

企业应用与行业软件

本月热点

●事件

宏达电将建立开放平台

宏达电CEO周永明向媒体表示，下一步计划打造开放平台，创建自己的应用程序商店，也计划将电信运营商的内容向平板电脑和智能手机推送。去年宏达电来自北美营收占比超过五成，欧洲占32%，亚洲及其他则仅17%。周永明认为，宏达电是国际品牌，未来市场布局最好是做到三分天下，但不会“一蹴而就”。

宏碁重返美国服务器市场

2月中旬，宏基在美国市场推出了一系列服务器和存储产品，这意味着宏基已重返美国市场。宏基此前曾在美国市场出售过服务器产品，但在过去的几年中，该项业务几乎销声匿迹，而这一次宏基表示将进行重大投资。宏基称，已拓展了支持及服务能力，并将在美国通过第三方制造商服务器。IDC数据显示，去年第四季度宏基是全球第三大PC厂商。在美国市场，宏基去年第四季度的市场份额为9%，位居惠普、戴尔和东芝之后。

近年来，在信息技术领域，云计算作为一种新兴的计算模式被提出，并迅速从概念走向应用。云计算的超大规模、虚拟化、多用户、高可靠性、高可扩展性等特点正是物联网规模化、智能化发展所需的技术。目前，物联网的应用领域正从面向企业的智能交通、电力抄表等发展到面向公众的个人医疗、智能家居等，遍及各行各业，但未大规模普及。Forrester预测，到2020年，世界上物物互联的业务跟人与人通信的业务相比，将达到30:1，因此，物联网被称为是下一个万亿级的通信业务。

物联网产业覆盖了传感感知、传输通道、运算处理、行业应用等领域，其中涉及的技术包括RFID射频识别、传感器、无线网络传输、高性能计算、智能控制等。云计算是将动态、易扩展且被虚拟化的计算资源通过互联网提供出来的一种服务。

我们可以将物联网抽象为感知单元、存储单元、传输单元、计算控制单元、展现单元组成的一个“超级计算机”，其中感知单元是“键盘、鼠标”，感知的不是人的点击，而是热、力、光、电、声、位移等信号；存储单元是“内存、硬盘”，在物联网中还包括嵌入到电子标签或传感网络节点中的存储设备，存储单元保存感知采集到的各种信息，如温度、湿度、电子标签编码等，以及这些信息的处理结果；传输单元是“总线”，将信息通过无线传感网、电信网传送到全球每一个角落；计算控制单元是“中央处理器CPU”，负责对信息进行集中、大规模快速处理和分析，发送指令，进行指挥调度；展现单元是“显示器”，通过电脑、手机或其他设备进行物联网信息的展现。

在物联网“超级计算机”抽象模型中，我们可以看出电信运营商扮演的是传输单元的角色，通过无线传输技术，将传感单元与计算控制单元、显示单元、存储单元关联起来。并且具有全球数据通信功能。

物联网运营平台显示的云计算特征，适合采用云计算技术建立基于云计算的物联网运营平台，其体系架构主要由以下几部分构成：云基础设施、云平台、云应用、云管理。物联网运营平台架构在云计算之上，既能够降低初期成本，又解决了未来物联网规模化发展过程中对海量数据的存储、计算需求。

与此同时，智能手持终端在网络应用模式的变化下，逐步走上了替代PC作为简易终端的道路，应运而生的iOS平台以及Android平台以其优秀的稳定性、 可扩展性迅速占领了移动智能应用的最高点，再加上智能终端硬件的迅猛发展，高清屏、多点触摸、重力感应、GPS应用等给更多的人们带来了全新的人机交互模式，让原来的鼠标、键盘走向了更为直观、高效、便捷的Mind+Finger模式。我们可以通过在移动智能终端上通过手指或者简单的甩动动作就能控制连上了云计算的设备，如空调、微波炉、生产车间的机械手臂。

非关系型数据库之Apache CouchDB

主持人：
俞黎敏，IBM高级工程师，满江红开放技术研究组织核心成员。开源爱好者，在各大技术社区为推动开源和敏捷开发积极摇旗呐喊、加油！

数据库

随着微博、SNS等Web 2.0网站大量涌现，非关系型数据库技术在这类网站中得到大量使用，也成为非常有吸引力的技术之一。这类网站，对数据库事务的要求并不非常严格，并且对读取一致性要求很低，而且在某些情况下对数据库写入的一致性要求也不高，这样就会造成数据库事务成为非常沉重的性能瓶颈。

对于传统应用程序里的关系型数据库，在应用程序写入一条记录之后，立刻进行查询操作，是肯定可以读出来这条记录信息的。但是对于Web 2.0的网站，数据库这种写入实时性和读取实时性需求并不高，因此，各种各样非关系数据库，特别以键—值存储的数据库发展得相当快，比如Redis、MongoDB、CouchDB等等。

Apache CouchDB是满足海量存储和访问、面向文档的非关系型数据库，采用Erlang语言开发，现在已经渡过Apache项目孵化器期，成为了Apache基金会的顶级项目。谈到面向文档就会让人想起大名鼎鼎的IBM Lotus数据库。CouchDB可以通过JavaScript采用MapReduce风格来进行查询和索引，还提供了双向冲突检测和增量复制的解决方案，提供RESTful JSON API以便在多种环境下采用统一的HTTP请求进行访问，这样就可以有很多第三方的客户端供开发者选择，简化了选择与开发的成本。CouchDB还内置了Web管理控制台，可以直接通过浏览器发出HTTP请求来与数据库进行相关的交互操作。CouchDB采用Erlang编写，功能强大的Erlang编程语言也是构建并发分布式系统的理想选择，很容易构建出高可伸缩性的和易于扩展的架构。

CouchDB的数据存储方式类似Lucene的索引文件格式，特别适合存储文档，很适合像内容管理系统之类的应用。CouchDB最大的意义在于它是一个面向Web应用的新一代存储系统，可以把存储系统分布到多台物理机器上面，并且很好地协调和同步各个节点之间的数据读写一致性，当然这归功于Erlang无与伦比的并发特性，对于基于Web的大规模文档应用，CouchDB的分布式可以不必像传统的关系数据库那样进行分库拆表，并且不必在应用代码层进行大量的改动。

一个CouchDB的文档是由许多字段组成的对象，这些字段可以是字符串、数字、时间，也可以是有序列表和关联Map，比如，一篇博客文章的全部内容。而一个CouchDB数据库正是这些文档内容的集合，每一个文档都有一个唯一的ID。CouchDB集成了“视图”技术，可以根据需求动态地创建多种视图，并且不影响基本的文档内容。CouchDB不像关系型数据库需要严格的Schema定义，它可以根据需要随意进行，因此不存在Schema新旧更新的问题。

在REST架构风格成为主流的时候，CouchDB仅仅提供了基于HTTP REST的接口，也许并不是另辟蹊径那么简单，它能否在更广阔的领域一展身手，值得我们期待。

本月热点

新品发布

Redis 2.2 RC4发布

Redis是一个高性能的开源Key-Value数据库。它的出现，很大程度补偿了Memcached这类Key-Value存储的不足，在部分场合可以对关系数据库起到很好的补充作用。支持保存List链表和Set集合的数据结构，还支持对List进行各种操作，例如从List两端Push和Pop数据、取List区间、排序等等，对Set支持各种集合的并集交集操作。它提供了Python、Ruby、Erlang、PHP客户端，还有一个高性能的Java客户端JRedis，用来连接到Redis数据库，提供同步和异步的连接。

MongoDB 1.7.5发布

MongoDB是一个介于关系数据库和非关系数据库之间的产品，是非关系数据库当中功能最丰富、最像关系数据库的。它支持的数据结构很松散，是类似JSON的BJSON格式，可以存储比较复杂的数据类型。

Apache CouchDB 1.0.2正式发布

CouchDB 是一个采用Erlang编写的面向文档的非关系数据库。提供了以 JSON 作为数据格式的REST接口来对其进行操作，可以通过视图来操纵文档的组织和呈现。

Seam 2.2.1 正式发布

这个版本主要集中在Bug修复和将JBoss AS 6作为部署环境。

支持使用JBoss AS 6部署所有的例子，Spring、DVD Store、Blog和JEE5-Booking，包含了针对于JBoss AS 6的脚本，可以针对JBoss AS 6进行打包和部署。如果在应用当中遇到问题，可以从这些例子中找到一些灵感。

主持人：
王翔，软件架构师，主要研究方向为XML、.NET、领域设计和PKI应用。工作之余喜爱旅游、写作和烹饪。

2011年数据库市场——不走寻常路

数据库

本月热点

●事件

甲骨文修补Java中存在十年的Bug

2月7日，甲骨文已经修复在Java框架中存在10年之久的漏洞，该漏洞允许黑客通过发送超长的小数位数攻击来降低服务器的敏感度。该漏洞源于二进制下一些浮点数的显示问题，当Java应用程序的处理数值为2.2250738585072012e-308时，该漏洞可能引起用户遭到拒绝服务的攻击。

甲骨文推出SOA套件

甲骨文公司日前推出了Oracle服务总线11g，甲骨文表示，此举增强了Oracle SOA套件11g在性能和可扩展性。Oracle服务总线11g将复杂和脆弱的应用架构转变成了灵活的、可轻松和快速修改的应用网络。该产品以一种一致的、基于标准的方法协调和管理服务及应用，并以此实现上述转变。

甲骨文发布首款数据库防火墙

2月16日，甲骨文发布了它的第一款数据库防火墙，该产品的能够保护有价值的系统免受攻击和骚扰。自从去年甲骨文收购Secerno之后，该公司一直在进行数据库防火墙（Database Firewall）使用技术的开发。该产品能够实时监控非授权侵入、SQL攻击。

●产品

Oracle Linux 6 正式版发布

在此版本中，甲骨文以Red Hat Linux做为起始，移除了Red Hat的商标，然后加入了Linux的错误修正。

2010年的数据库市场处处涌动着湍流，不管是传统意义上的数据库“正规军”还是崛起于Web 2.0的“杂牌军”都在标新立异。

往年的数据库市场热点不外是“你升、我升、大家升”的市场推广，2010年却相反，最大的风头被NoSQL抢走了。按照我们中国人传统的思维——“木秀于林，风必摧之”，但IT业似乎正好相反，“木秀于林——风必‘吹’之”，即一旦出现某个新概念就会被各方推手“吹”起一股风潮，数据库领域更是如此。在国内，尽管NoSQL的实际应用寥寥，尤其是把NoSQL用于自家核心“生命线”项目的更堪称“罕见”，但这似乎一点也不影响NoSQL大行其道。究其原因，这个趋势很大程度上来自技术实用主义。既然NoSQL看上去这么美，为什么落地却如此阳春白雪呢？究其原因在于缺少能够驾驭它的好骑手。此外，从几家美国NoSQL厂商近期公布的技术资料看，NoSQL似乎在走曲线。尽管下层物理实现各有特点，但上层提供给开发者的手段却多少有点回归SQL的味道。凭此我们也许可期待在不远的将来，NoSQL会成为数据库技术实现层面的一次革新“插曲”，而最终开发人员面临的还是半OO、半关系的类SQL。

数据库一体机在2011年将是焦点，尤其对部分“海量级”处理的企业应用而言，一体机的魅力显而易见。不过冷静回味一下，一体机的概念大家颇为熟悉，有炒冷饭的意味，因为我们20多年前用的很多小型机就是这样的，只不过那时候的计算能力有限，没有把那么多Fellow的经验浓缩为Best Practice和Guideline后植入一体机。为什么20多年后，这个似曾相识的产品能再次获得关注呢？恐怕来自我们自身。数据库技术人员也免不了矛盾的一面，一方面希望集成的产品开放，商务上、技术上都能主动；另一方面真正使用之后又希望能“一口对外”，锁定一个下家做好技术支持，或者希望自己做的东西其下层平台足够稳定、足够聪明，让自己省心。一体机的出现给了后者不错的选择，不过成本是门槛，因此很多用户都在观望，看看走在前面的国内同行“螃蟹”吃得如何，如果效果不错相信会对以后的方案评审会产生影响。现在我们评审项目时在采购部分往往会审核用了多少何种配置的服务器、多少个数据库许可、多少存储，而以后也许会以一体机为计数单位。**地理信息数据库**也是个趋势。现在我们的数据库多数面向事务性，而随着移动应用、车载应用、物联网应用的兴起，地理信息数据成了联系我们和物理世界的纽扣，以至各主要商用数据库厂商纷纷将地理数据作为一个“一等公民”般的数据类型纳入其中。

2010年匆匆而过，从很多渠道的反馈看，今年的数据库市场仍将充满个性，后经济危机时代的资本溢出效应相信会为数据库市场带来更多惊喜。2011年将是数据库技术破茧而出的一年。

“MicroKia”来了！

主持人：
涂曙光，微软平台技术专家，专注于.NET、Web、SharePoint、Office等技术领域。现任职于中国惠普有限公司。

.NET

虽然之前就有很多传闻，但是当微软和诺基亚正式宣布了它们将在手机领域所进行的各种合作时，仍然让我们感到些许吃惊。

在今天日益扩大的智能手机市场中，面对着苹果的iPhone与Google的Android，诺基亚和微软看起来就像一对“失意者”。诺基亚的Symbian系统和微软的Windows Mobile都曾经是智能手机市场上举足轻重的角色，但显然它们在今天已经失去了往日的领先地位，几乎没有人会相信Symbian和Windows Mobile会在未来的智能手机市场上能与iPhone和Android一较高下。微软痛定思痛，干脆把Windows Mobile系列推翻之后重新做出了Windows Phone 7。但当今的局面却是iPhone和Android手机已经满天飞，Windows Phone 7虽好，但如果没有足够数量的设备供消费者选择，它注定没法成为一个主流的平台。而诺基亚的情况则刚好相反，虽然诺基亚仍然占据着庞大的市场份额并且拥有许多拥趸，但Symbian系统的用户体验始终无法与iPhone与Android相比，诺基亚之前曾寄予厚望的MeeGo进展缓慢，诺基亚需要的是一个成熟的、能被装进其各式各样终端设备中的、用户体验足够好的手机操作系统。

微软与诺基亚未来的合作方向包括，诺基亚将使用Windows Phone作为其智能手机的主要操作系统，诺基亚也会协助并定义Windows Phone的未来发展。这样的发展方向，无疑针对的就是微软与诺基亚各自薄弱的环节，利用诺基亚的硬件制造技术与在手机市场的经验，再加上微软的Windows Phone，产生1+1>2的结果。

笔者对微软与诺基亚在手机领域的合作比较看好，相信这将形成与iOS、Android三足鼎立的局面。对微软开发社区的开发人员，这也是一个利好的消息，因为这意味着.NET开发人员，特别是Silverlight开发人员，将也有机会在移动开发领域大有作为。

除了有关Windows Phone的这个重大消息之外，微软在Web领域也发布了两个重要的产品：Internet Explorer 9 RC和ASP.NET MVC 3 RTM。

笔者亲自安装并试用了IE9 RC，简洁的界面、不错的速度、新颖而不会干扰用户的新提示框，都让IE9的使用过程相当愉悦，让笔者感到这才是一个好用的浏览器应该有的样子。在Web页面的设计已经日趋遵循标准化的今天，笔者相信用户们向IE9的升级速度，会远远快于IE7/8，因为向IE9的升级几乎不会造成网站访问性能上的问题。在尝试IE9的过程中，令笔者最感兴趣的特性之一就是它与Windows 7的整合性。当浏览某个网站时，用户可以直接将当前标签页拖动到Windows的任务栏上，让这个网站如同一个应用程序一般“钉”在任务栏上，以后用户就可以像启动一个应用程序一样，通过点击任务栏上的图标，直接打开这个Web网站。这不就是“Web Apps”的最佳体现吗？

注：MicroKia只是坊间对Microsoft+Nokia的简称，未获任何官方确认。

本月热点

● 新品发布

ASP.NET MVC 3 RTM

ASP.NET MVC 发布了第三个版本。ASP.NET MVC通过自身的简洁、开源已经吸引了大量的ASP.NET开发人员。

ASP.NET MVC的官方网站是http://www.asp.net/mvc。

● 资源

First look at 5 features of IE9 RC

来自IE开发团队的Ari Bixhorn通过这个视频，向观众展示了IE9 RC中的5个涉及到用户体验方面的新特性。

视频网址：http://channel9.msdn.com/posts/First-look-at-5-features-of-IE9-RC。

mvcConf

mvcConf是一个在线的ASP.NET MVC技术教学节目，它向观众介绍了最新发布的ASP.NET MVC 3的最新内容，以及其他与Web开发有关系的技术讲座。

课程网址：http://channel9.msdn.com/Series/mvcConf。

SharePoint 2010 Firestarter

SharePoint 2010 Firestarter是一个为期一天的面向SharePoint开发人员的教学课程，它介绍了SharePoint 2010的开发特性，涵盖了Visual Studio 2010中有关SharePoint工具的部分，并介绍了基于SharePoint Online平台的开发。

课程网址：http://channel9.msdn.com/Series/SharePoint-2010-Firestarter。

主持人：
钱宏武，现任职盛大创新院，原搜狐互动产品开发部主管，资深互联网社区架构师，垂直搜索领域专家。

改变，从自己开始

Web

本月热点

● 事件

Varnish 2.1.5 正式版发布

号称最快的HTTP代理服务，如果能更稳定，将是最合适的选择。

Facebook身价达829亿美元

Facebook身价，超过亚马逊位居第二。我也看过SNS最早的理论，可扎克伯格照着这个理论做了。这个就是理论家和实干家的差别：数百亿的市值。

HTML5发布新Logo

HTML5的缔造者WHATWG，先是换了一个Logo，一天后，又作出了一项决定，今后对其维护的标准将只称为HTML，不再使用5这个版本号。这个决定是一种理解的深入，或者叫做一种思路的变化。开始都认为HTML5和HTML4一样，是一种标签语言的规范，但随着HTML5不断扩充，它实际代表了新的富互联网应用开发方式，涵盖了HTML（WHATWG）、CSS、JavaScript、SVG、WOFF等技术标准，它已经不像是HTML4的一个简单的版本升级了。

对于HTML5、IPv6，这种未来互联网的新标准、新底层，将孕育多少令人瞠目的新应用和公司呢？令人期待。

VeryCD关停了音乐和影视两大下载服务，这两个服务的流量加起来占到VeryCD网站总流量的一半以上。此次关停对创始人是一个艰难的决定，对于我这样资深的用户来说，一下子也确实很难接受，毕竟习惯了从那里下载歌曲和电影。

但现在想想，该来还是要来的。初期互联网靠免费，迅速培养了大家的消费习惯、思维习惯。被误导的消费观念让我们对于无形产品，如建议和方法、软件、影音娱乐等都不是太珍惜，认为都应该免费。很多人对于免费振振有词，“互联网的精神就是分享”云云。

自买了iPhone4后，很多人劝我越狱，因为能免费使用收费软件。后来想想，用得了那么多的软件吗？最常用的就那么几个，其中不少都有免费的版本。去折腾，花了这么多的时间和精力，能省多少钱？而付了费，这些精力和时间，可以花在更有价值的事情上。何况你还是一个程序员，自己都不珍惜程序员的劳动，谁还会珍惜了？

想想那些横行的流氓软件吧，是谁把这些充满顶尖智慧的东西推向了恶魔？除了消费者，我们程序员自己是不是也在破解、免费这种变态的逻辑中，成了恶魔的帮凶？而想转变这些让我们自己愤恨的环境：软件不值钱，服务不值钱，唯一能做的，就是从自己开始。于是VeryCD关掉了部分的下载服务，方向是正确的。

这一切只是开始，我们的互联网环境相对比较差，怨天尤人从来没有用，从自己开始改变，尝试着用你网上的第一个付费的软件、第一首付费的歌曲、第一本付费小说 、第一部付费下载电影。这些都不是负担很大的事情，但每做一点，都在改变这个较差的环境。是把我们这个赖以生存的虚拟世界推向一个泥潭，还是把它改变到一个相对有序、清晰的环境，这都依赖于我们的行动。

和这些相比，最让人害怕的就是IPv4地址即将用尽的问题。在2010年初，就有观点预言过IP地址将用尽的现象，当时国内某机构曾提出，IP地址将在2011年8月份用尽，而文顿·瑟夫的新预言又指出地址耗尽的速度正在加快。对于满天飞的消息，很多普通用户担心用尽后，无法申请上网还情有可原，可有些程序员还在担心，就实在让人无语了。IP地址一旦分配完毕，确实有新加的用户无法申请到IP的问题，但这从来都不是一个坏事，因为IPv6的技术已经搁置多年，在IPv4用尽后，IPv6一定会普及开去，同时很多新的技术，如HTML5等，会随着IPv6更新迅速普及。所以我们正好可随着这次底层技术的变革，用上更好、更稳定的新服务。虽然代价是有点大，但哪次变革不需要代价？想想未来我们用IPv6操控一切的时代，下班前通过互联网控制电饭煲做饭，浴缸放水，路上通过手机看看水的温度，这些以前科幻小说中的情节，随着IPv6普及，实现起来将更加容易、成本将更低，更多的人将受惠。

多种开发语言混合开发游戏项目

游戏开发

主持人：
宋忆疆，2000年开始从事游戏程序开发，参与的项目《碧血情天》，《傲世三国2》，《乱舞天下》，《流星蝴蝶剑OL》等的研发。目前担任《流星蝴蝶剑OL》项目制作人。

游戏的过去、现在，甚至可以预期的将来，C/C++语言都是开发的重心。但是随着游戏结构日益复杂，C/C++过于复杂、难以掌控的问题逐渐浮出水面，而现代计算机处理能力的日渐提升，给一些高级语言介入游戏开发，提供了比较坚实的基础。

让我们先看看在游戏开发中，大家比较常用的语言。

ASM：汇编语言目前看来在游戏开发中所占比例已经越来越低，除开一些极其需要性能的底层库偶尔会用到，很少有人用它来做逻辑结构处理。当然一些新的SIMD指令技术在一些特殊处理上，还是有些市场的。

C/C++：这是核心开发语言，性能优秀、语法灵活、功能非常强大，绝大部分游戏的核心代码都可实现。不过C/C++的优点，也是缺点，过于灵活的语言和强大的系统控制能力，对开发人员来说，是一个需要随时保持警惕的妖怪。再优秀的程序员也可能在上面栽跟头，如果你团队里面有短板，那么很可能让整个项目买单。

Python、Lua一类的解释执行脚本语言，在现在的游戏开发中，已经占有很重要的阵地，几乎绝大部分游戏，都会或多或少用到解释执行的脚本。作为脚本语言，语法都会比较简单，而且很难犯错，即使脚本运行时出现异常，也能把脚本终止，不会对整个系统带来太大的危害。而且动态语言修改后，不需要做太多编译处理，适合开发过程中边改边测试。很多团队在开发项目的时候，脚本甚至都交给了非程序开发人员编写。有一些引擎，会定制出自己的脚本，比如UE3的UC脚本，在语言层就可以内嵌对引擎的支持，使用起来在某些方面会更方便。几乎所有的脚本语言，都提供了本地立即代码和解释代码之间的相互调用，这样在游戏开发中，我们可以将某些事件状态预留给脚本实现。在脚本实现的过程中，可以调用本地代码的功能接口。本地代码和解释代码之间的绑定，是一个比较有意思的话题，怎么做到非常方便而高效率的绑定，每个语言都有自己的一些方案。C++代码和脚本之间的绑定，模板技术是关键，比如绑定Lua的LuaBind一类的库，都是基于模板实现的。

.NET：其实这不能说是某个语言了，微软提供了一个.NET框架，巧妙地把编译分成了前端编译、后端编译。编译器通过前端编译，把文本文件编译成为微软中指令(MSIL)，然后运行在所有Windows平台上的.NET框架会在执行程序的时候，把中间代码通过后端编译器，编译成真正的本地立即代码（这个过程被称为JIT）。微软可以把类C++风格的CLI、C#、Basic，甚至Python统一编译成为MSIL，程序执行的时候JIT成为汇编代码在目标平台运行。

.NET和C++混合编程，将是今后Windows游戏开发的重要方向。C++实现底层要求高效率的代码，而.NET平台Reflection、GC和丰富的类库，是项目逻辑开发效率提升的源泉。

本月热点

● 事件

微软、NV退出PC游戏联盟

记得业界多家厂商在2008年初成立了声势浩大的“PC游戏联盟”（PCGA），不过一直没什么成效，反倒是负面消息不断，不断有大碗与之决裂。重量级发行商Actvision在联盟成立一年多后就脱离了组织，如今微软、NVIDIA也宣布退出。

盛大游戏将推出《七龙珠OL》

盛大游戏董事长兼CEO谭群钊透露，盛大游戏将在2011年中推出《七龙珠OL》，希望《七龙珠OL》能给看这部漫画书长大的玩家带来更多乐趣。

《九阴真经》不再跳票

《九阴真经》经过多次跳票后，面对玩家的期待，苏州蜗牛表示《九阴真经》不会再跳票，希望在2011年中占领武侠网游的最高阵地。

● 会议

2010年度中国游戏产业年会

2010年度中国游戏产业年会于2011年1月19日在北京市石景山召开，各位行业大佬云集，会议评选了包括最受欢迎游戏、最佳媒体等奖项的2010游戏产业“金凤凰”奖。

主持人：
马宁，微软最有价值专家，Windows Mobile开发者。

猜想 2011

新平台

本月热点

●新品发布

SONY发布新一代游戏掌机NGP

1月27日，广大游戏迷期盼已久的PSP2终于发布了，不过名字改成了NGP。采用了四核的ARM Cortex A9的处理器，分辨率为960x544的5寸OLED屏幕，支持触摸屏、前后镜头、双游戏摇杆、六轴传感器，网络方面支持3G、WIFI和蓝牙。

在任天堂发布3DS之后，正如NGP的名字一样，NGP代表了下一代的游戏掌机的发展方向。由于iPad在游戏方面的出色表现，触摸屏和重力感应已经成为游戏的标配，并且衍生出很多有趣的游戏形式。最有趣的是，NGP增加了双摇杆，这说明SONY仍然坚持以摇杆和按键为主要输入方式，但双摇杆是不是意味着会产生一些新的游戏方式呢？

其实NGP最大的改变是放弃了UMD。这背后隐藏的是盈利模式的改变。UMD是PSP游戏的载体，第三方厂商通过销售UMD获利。而取消UMD之后，下载可能成为NGP游戏发布的主要渠道。而且SONY同时推出了PlayStation Suite，支持Android游戏运行在PS3上。看来对于SONY来说，模式创新比产品创新更加重要。

苹果App Store下载量达到100亿

我们必须承认App Store创造了移动互联网时代的商业奇迹。App Store将以往由大型软件公司把持的软件零售渠道扩展到个人和小团队开发者，使用户能够用更便宜的价格获取软件，并且极大简化了购买、发布的步骤。当然，App Store的商业奇迹不是建立在旷野之上的，iTunes就已经是最成功的在线音乐商店了。App Store只是将发布的内容从音乐扩展到了应用而已。

时间进入2011年，全球的互联网行业都处于一种躁动中，所有人都预感到一个大时代即将到来，但是没人能够说出这个时代将会是什么样子。我们试图用猜想的方式，来描绘一下2011年移动互联网行业可能出现的新变化。

Game Center成为最火的移动应用？

Game Center也许会成为下一个移动互联网的爆发点。Game Center是iPhone的叫法，在微软叫Xbox Live，在Sony叫PSN，总之，Game Center是移动游戏的交互中心。除了这些厂商提供的服务外，OpenFeint和Plus+则是由第三方提供的游戏社交服务。与App Store必须要有操作系统支持不同，厂商提供的Game Center有时候反而不如第三方的服务。Game Center具有更强的社交性，不能指望你的朋友都用相同的手机，跨平台的Game Center反而会获得青睐。Game Center只有增强移动互联、位置服务、多人在线的功能后，Game Center才能够大放异彩。

谁还会加入平板战场？

CES上满世界都是平板，如同“下一个谁会来的游戏”进入了高潮部分。让我们猜猜下一个敲门的会是谁吧？首先是webOS，自从下嫁了惠普，智能手机方向就再也听不到枪声了，平板市场下面挖坑道的声音却越来越近。没办法，谁叫新夫君是做电脑起家的呢。可能性90%。

微软长久的问题就是PC和Mobile Device之间的路线斗争，平板也是如此，采用Windows，还是Windows Phone，这是个问题。平板的用户体验更接近智能手机，从Windows Phone升级而来难度不大。可是Windows团队是否甘心让出自己的地盘，这就要考验老大的智慧了。可能性60%。

Chrome OS，在Android阴影里茁壮成长的一株小草。从用户体验上来说，Chrome OS就是为上网本、平板设计的操作系统，但离不开网络的特点，这让Chrome OS显得过于超前。不过在某些企业级的领域，Chrome OS还是有安全性、维护性方面的便利。可能性30%。

Windows Phone 7会进中国吗？

现在似乎没人关心这个问题，除了微软公司。首先是运营商不积极，所有人的眼睛都盯着iPhone。其次是在线服务，很多对于中国用户来说就是一堆404，替换本地服务的成本颇高，且后台是否兼容也是一个大问题。最后，缺乏本土化厂商的支持。其实，微软应该找联想去合作，LePhone目前销量不佳，同事联想缺乏的是在线服务和开发者的支持，而微软缺乏的是本土营销团队和售后体系支持。况且联想一直在做Windows Mobile的终端，不瘟不火、不离不弃。关键是，联想肯不肯把头从LePhone的绳圈里拿开，而微软能否敏锐地认识到中国市场的宝贵。

好戏开场，让我们拭目以待。

移动时代开发语言概述

主持人：
崔海斌，十几年的开发经验，专注于Java、Android、移动开发等领域。现于创新工场旗下的点心团队就任总架构师。

移动

目前的移动平台，最流行并且相对成熟的要数iOS、Android、Symbian、BlackBerry和WP7。前两个平台上，出现了很多应用开发者的成功案例，它们的应用程序数量分别达到了30万和20万，并在持续增加。我们先说说这两个平台上的开发语言。

iOS上的开发语言首选是Objective-C，配合Xcode、Interface Builder这两个开发利器以及高效的模拟器（支持OpenGL ES的开发），开发者很容易进行iOS应用的开发及调试。iOS也同时支持C/C++语言，无论是使用toolchain或者配合Objective-C进行开发都非常方便。除此以外，从2010年9月份开始，苹果放宽了iOS上的软件开发限制，允许使用各种开发语言进行iOS应用的开发，这中间最大的获益者，应该是.NET的开发人员，通过MonoTouch，.NET开发人员可以使用C#或者Delphi Prism开发iOS上的应用。

Android上首选的开发语言当然是Java，配合独特的Dalvik VM，广大Java开发人员可以在熟悉的Eclipse的IDE上使用ADT插件轻松进行Android应用的开发和调试（在此要说明的是，Android的模拟器效率极其低下，比起iOS的模拟器来说，简直是天上地下，不过Google在最新的Honeycomb SDK发布声明中承诺会尽快改善此问题）。Android上保留了Java的GC和众多优秀的类库，也继承了性能低下及消耗内存过大导致GC运行效率低下的问题，因此，在性能要求高的Android应用中，推荐大家使用C/C++在NDK下配合Java通过JNI进行开发。而从Gingerbread开始，Android甚至开始支持单纯使用C/C++进行应用开发，未来会有更多优秀的应用和游戏移植到Android上。此外，在最新的Honeycomb上，Android引入了应用的硬件加速机制，只要简单声明你所开发的应用会使用硬件加速，Android就会自动使用硬件的GPU进行图形加速显示。Android也支持包括Mono支持下的.NET开发语言以及一些脚本语言如Python等。RenderScript是Android平台上为解决高性能绘图和动画所开发的类似C语言的一个脚本语言，通过它很容易地开发出高性能的2D和3D应用，而不需要考虑OpenGL ES的具体版本和硬件支持，一切将在设备端动态编译成最优的代码。

无论未来Nokia在Symbian和Meego上投入如何，Qt都是最重要的开发平台。Qt上主要支持C++。

WP7上的开发语言首选当然是.NET平台下的语言，如C#、VB.NET和Delphi Prism等，框架使用的是Silverlight和XNA框架，开发人员如果熟悉了这套开发架构，还可以开发在Xbox 360上运行的应用和游戏。

总体来说，C/C++的开发人员在未来几年内移动平台不能解决性能、成本和耗电问题的情况下，还将是高性能移动开发的重要力量，Java开发人员也同样有多平台可以选择，需要重点关注的是，基于HTML5的Web应用，会在越来越多的平台上普及。

本月热点

● 事件

移动市场苹果份额下降

美国市场研究公司IHS旗下iSuppli于2月15日发布2010年全球移动应用商店营收报告，显示在市场出现了大量针对iPhone的竞争者之后，虽然Apple App Store仍然保持了统治地位，但BlackBerry App World、Nokia Ovi Store和Google Android Market的增长更快，夺取了较多市场份额。

报告表示，2010年免费移动应用依靠应用内购买等形式获取的总收入占24%左右，由于应用数量不断增多，更多开发者将采用免费策略，预计2011年北美市场的收入有一半来自免费应用。这将超过收费游戏的份额，后者在2010年占总额的52.2%。

谷歌Chrome网络商店支持世界各地开发者上传

谷歌正在迈出走向全球网络商店的第一步，2月，它已经允许各地的开发者为本地市场上传应用程序，并支付一定的酬金。谷歌软件工程师Qian Huang称，自从公司面向美国市场发布该网络商店以来，公司的开发者一直在努力将商店国际化。目前，当开发者上传程序时，可以看到很多国家选项，例如阿根廷，澳大利亚，巴西等。开发者能够根据用户的国籍来为应用程序定价，但美国本土之外的开发者不能获得报酬。

多盟移动广告平台发布

国内移动广告领域又迎来一位新的竞争者——多盟。多盟移动广告系统将于3月份正式对外发布。目前，多盟已有包括网易、天天动听、墨迹天气等近百个合作伙伴。运营VP边嘉耕表示，将推出一项面向移动创业者的服务，通过在多盟平台上整合（基于流量换算）并智能分配每个应用的相互推荐资源，形成一个良性的推荐联盟。

主持人：

张泉（花名伯飞），目前任职于淘宝网UED前端开发工程师，专注于JavaScipt及基相关技术。

前端 MVC 的应用

Web前端

本月热点

●新品发布

IE9 RC版发布

2月11日，Internet Explorer 9 RC版发布，将支持40种语言，RC版的发布意味着正式版已蓄势待发。RC版的新功能包括“跟踪保护”功能，用户可以选择是否允许第三方跟踪其在线活动；新的ActiveX过滤选项，允许用户关闭或开启ActiveX 插件；微软终于解决了用户对新版本最大的抱怨（有17000多条反馈），如果你想让标签和地址栏之间有一个分割线，RC版提供了选项，在浏览器顶部右键菜单开启；RC版的速度也快于Beta版，大部分的网页渲染问题已得到解决。

jQuery 1.5 发布

1月31日，jQuery 1.5 发布，此版本重写了Ajax 模块，提供了高级的统一API，新增延缓对象（Deferred Objects），可以直接使用没有立即返回的返回值，比如异步AJAX请求的返回结果，附加多个事件处理器。

●事件

Google全面转向WebM

Google宣布Chrome浏览器不支持H.264，转向WebM，是近期互联网业界一个不小的震动，同时这一决定和影响也将是深远的。

随着Ajax流行，JavaScript应用程序变得越来越复杂，特别是HTML5的兴起，以及Canvas、Video、Storage、Web Socket等很多强大功能的提出，JavaScript应用程序的规模必将越来越大。所以设计可重用、可维护的JavaScript应用程序正在变得越来越重要。

很多流行的JavaScript库，像jQuery、YUI，很大程度上提升了前端开发者的效率。然而面对一个复杂的JavaScript应用程序，要达到可维护性、可重用性，满足合团队协作开发的要求，仅有这些库还远远不够！与此相反，后端架构比较成熟，在各种优秀MVC框架的支持下，开发和维护Web应用变得更加简单。

现在JavaScript应用开发中主要有三个方面的任务，分别是操作DOM、侦听事件、操作数据。从jQuery、YUI我们也可以看到，这些库的核心都会有三个对象DOM、EVENT和AJAX。这些特征里我们已经看到了分离的概念。很多先行者也已经将这种分离的概念上升为框架级别，引入了MVC模式，并进行了很多实践，如 SproutCore、PureMVC、JavaScript MVC等。这些框架帮助开发者管理JavaScript代码和架构前端应用。

Model：数据模型用于封装与应用程序业务逻辑相关的数据，以及对数据的处理方法。数据模型管理应用程序数据状态，处理CRUD操作，响应数据查询，对数据进行验证。模型中的数据变化可通过观察者模式进行公布。在JavaScript应用中，用JSON代表数据，用AJAX存取数据。现在很多JavaScript库像Dojo、Ext里都有专门的Data包。不管数据是用AJAX获取，还是从本地存储或DOM属性数据中获取，都可以通过这个Data轻易操作。如果使用类CouchDB的基于文档的数据库，对前端数据层进行分离和封装非常必要的。

View：视图层调用Model数据渲染用户界面。为了实现视图的功能，视图需要监听数据模型的状态变化。在JavaScript应用视图层就是DOM，但是JavaScript应用视图层不只是要渲染Model数据，还要处理DOM元素，完善一些效果的展示。

Controller：控制器起到不同层面间的组织作用，用于控制应用程序的流程，它处理事件并作出响应。事件包括用户的行为和Model数据的改变。JavaScript应用是一个事件驱动的系统，控制器就是响应用户交互动作，并调用对应的事件处理函数。在这个处理函数中可能会调用Model更新数据，也可能会选择View渲染HTML。对于一个页面来说，可以只有一个Controller管理事件，也可以由多个Controller共同管理事件.

虽然MVC在JavaScript应用中已经有很多实例和阐述，但是MVC在JavaScript应用如何使用，还需要更多优秀的JavaScript框架去尝试。随着JavaScript应用规模的不断扩大，各种JavaScript框架会涌现，其中必定有很多模式值得借鉴和讨论。

安全厂商遭遇被开源

主持人：
肖新光，网名江海客，安天实验室首席技术架构师，研究方向为反病毒和计算机犯罪取证等。

安全

近期最令人震惊的消息莫过于卡巴斯基的源码泄露事件。一名前卡巴斯基工程师，因在2008年盗取了当时相对完整的卡巴桌面产品代码，并盗卖，被发现后获刑三年。目前这套代码已经开始在网络中私下流传。

从流传的代码来看，其中有相对完整的外围代码，甚至包含了部分产品化的外围设计文档，只是并不包含真正的卡巴引擎的核心模块代码（这些模块是有关于虚拟机、启发式、脱壳等部分，被视为卡巴技术能力的精华）。但泄露的这套代码也有助于大家编写调用这些机制的一些外围的调度代码。

实际上，作为重载反病毒引擎的代表，卡巴斯基已经被业界以及民间爱好者非常深入地分析过。卡巴斯基所实现的反病毒引擎模块库记录化设计，是其引擎设计中的精华所在，而其核心数据结构已经被剖析得比较清晰了。甚至在上世纪90年代末期就出现过一些直接盗用卡巴斯基OBJ模块使用的情况。为此，卡巴斯基也采用了一些反制措施，包括在OBJ模块中加入暗记、逐步把AVC格式替换为KDC格式等等，而更重要的是法律手段的跟进，使相关情况日趋减少。

尽管卡巴引擎对终端系统来说，有很好的优势和特色，资深AVER也都比较了解其结构，但基本上没有哪个主流产品能够完整地沿着学习卡巴斯基的道路来走。

反病毒是一种体系的对抗，一般来说，越是先进的反病毒引擎，其维护压力越大。后进厂商虽然投入重金，也多半只能采用所谓云查杀等全HASH检测方法，维护一个高质量的基于代码缓冲区和特征检测的本地引擎，依然很有困难。各个传统厂商在前进过程中，均已经形成了一套适合自身运维能力的维护体制，这个体制有很大的继承性，很难推倒重来。

随着与海量病毒的对抗不断升级，反病毒引擎与产品都只是冰山一角，而更加庞大的水下冰山部分，则是其后台分析维护体系和整个团队。而后者更难于模仿抄袭。

因此，尽管有可能出现个别的山寨小厂商，但应该对卡巴斯基的核心竞争能力并无显著威胁。但以国内的一些安全项目特点，出现一些山寨卡巴斯基型的“科研成果”，似乎将是必然的。因为相关的命运曾发生于包括snort、nessus在内的所有较大的开源项目上，这些开源项目的汉化被冠以自主知识产权之名而试图蒙混过关。今天，恐怕同样的事情也会发生在这套泄露的商用产品之上。

这套代码遭到源码级的漏洞发掘，是未来可能面对的问题。但反病毒比较容易出现的问题多数在其格式解析和预处理部分，它们在卡巴斯基目前尚未被发现泄露的关键代码中，因此用户不必太过惊慌，只是要小心假冒卡巴斯基产品的下载。

正如部分从业人员所担心的，这个事件也许会让安全厂商内部管理变得过于苛刻和保守，而不利于反病毒事业的发展。

本月热点

● 会议

RSA展会

被称为网络安全年度盛会的RSA展会于2月14日在旧金山召开，国内厂商华赛、绿盟和江南科友均独立出展，而中关村则组织启明星辰、网御神州、天融信、安天等七家公司联合出展。

赛门铁克推免费试用版扩展验证SSL证书

赛门铁克公司日前推出免费试用版的所有威瑞信SSL证书，包括业界首个免费试用版的全功能扩展验证(EV)SSL。现在，网站所有者和运营商能够在30天内，在实时网站中免费试用任一威瑞信SSL证书。赛门铁克还允许不要求SSL加密的企业免费试用威瑞信Trust Seal 60天。

Windows系统MHTML协议中存高危0day漏洞

微软近日证实Windows系统MHTML协议中存在一个高危0day漏洞，可能导致用户密码、电子邮件等重要信息泄露。微软官方随即提供了临时补丁，但因该补丁与Google等国内外知名网站存在较大兼容性问题，致使已安装网民中约有三分之一的人又撤销了该补丁。

报告称中国黑客入侵全球能源公司计算机网络

迈克菲公司公布了名为《Global Energy Cyberattacks: “Night Dragon”》的报告，称从2009年11月开始，西方石油、能源和天然气公司遭到了秘密协调的有针对性的网络攻击，黑客利用社会工程、钓鱼手段和微软Windows漏洞、Microsoft Active Directory弱点和远程管理系统，窃取能源公司在运营、项目融资、油气开采和油田投标方面的敏感信息。

不只是 .NET

——记nBazzar技术会议

■ 文 / 佳琪

2011年1月16日，在上海举行的第三届nBazaar技术交流会，吸引了上海及附近省市近200名技术人员参加。本次活动的组织者，来自盛大创新院的赵劼表示，扩大技术人员的眼界，关注.NET技术人员，而不仅仅是.NET技术，是他举办nBazaar技术交流会的原则及核心目的。

演讲内容精彩纷呈

与过去两届活动相同，第三届nBazaar技术交流会也准备了四场演讲。

第一场演讲的主题是《针对iPad平台的高性能网站架构》，演讲者是英孚教育的Tech Leader，马士杰。

摩根士丹利发布的《移动互联网报告》认为移动互联网周期是50年来的第5个新技术周期，以Apple的iPad平板电脑和手机上网为代表的移动互联网的增长势头将超过电脑上网，这也对传统Web开发方式及技术架构提出了新的要求。

对此，马士杰认为，首先，他认为在传统开发常见的客户端优化进一步成为重中之重。此外，iPad也带来了新的浏览方式，例如传统的鼠标移动在iPad则是以“Touch”来表示，而Tap等操作也与传统的鼠标点击有所出入。

在演讲中，马士杰和听众分享了他近一年在针对iPad平台的高性能网站架构方面的一些经验，例如如何兼容不同平台桌面和移动浏览器的表现层设计模式、如何针对iPad Mobile Safari浏览器的Web页面进行性能优化、如何对iPad本地程序和在线网站的无缝整合等。

第二场演讲的主题《分布式版本管理》与.NET技术并无直接联系，但它却是每个技术人员都无法回避的问题。

演讲者李骏谈到，经过数十年的发展，CVS、SVN等上一代版本管理系统中的一些问题，催生着新的技术及其应用模式。在过去几年中以Git和Mercurial为代表的分布式版本管理工具有较大的发展，已经基本具备了普及应用的基础。李骏介绍了分布式版本管理欲解决的问题及其关键价值，并以Mercurial为例介绍具体使用的方法和流程。他清晰细致地讲解了Mercurial的每一步操作所产生的变化，从听众在现场的微博墙上的反馈来看，这场演讲的效果十分令人满意。

第三场演讲者是来自SAP的师建苗，主题是《企业开发领域的语言特性》。

师建苗表示，他跟着谭浩强的C语言课本和Borland的TC 2.0进入编程领域，后来又接触了C++、Delphi、Python、C#、Java等语言，感觉语言并非那么重要。但2007年加入SAP Labs China，从事GRC软件产品研发之后，他忽然认识到，对于优秀的软件，一些针对性语言特性还是有所帮助的。

在演讲中，师建苗谈论了对于企业应用来说非常重要的几个方面，并阐述了ABAP这门面向企业应用的语言是如何有针对性地应对这些方面的。这场演讲为许多平时只接触过C#、Java等通用编程语言的技术人员展示了另一片天地。

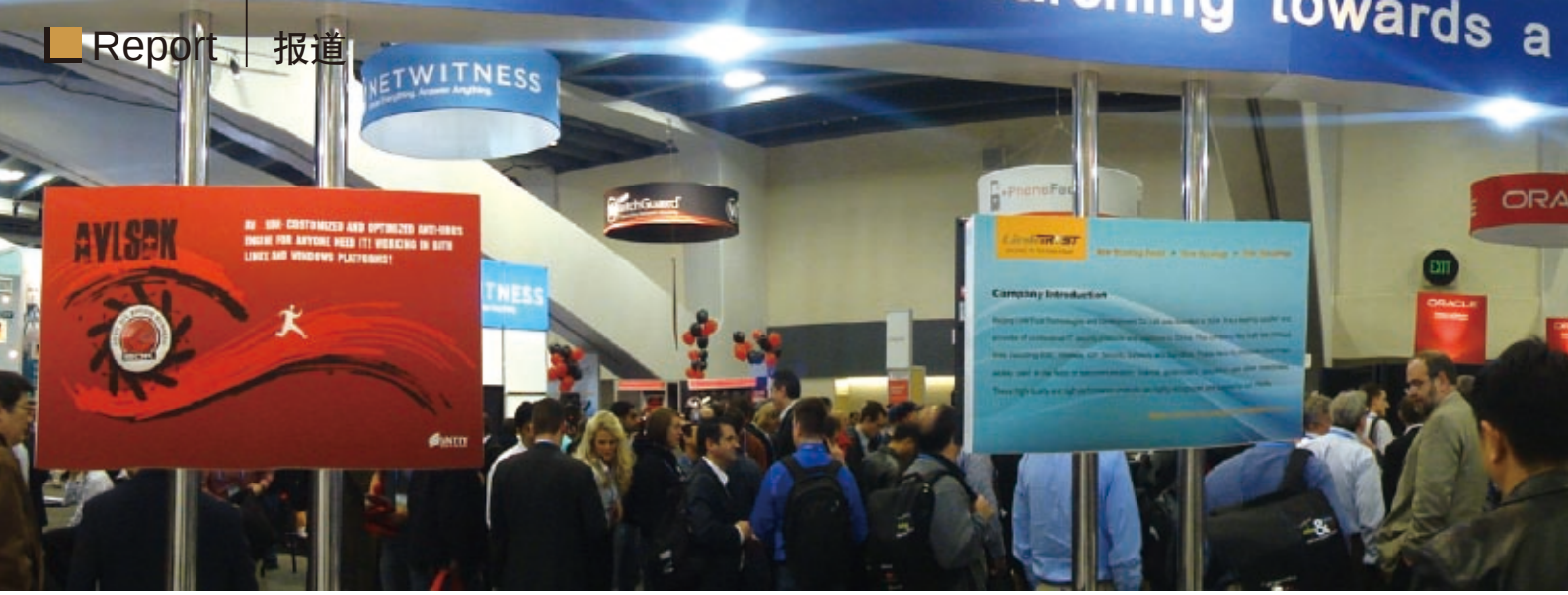
第四场演讲是《使用.NET构建轻量级分布式框架》，其演讲者乔捷是瓦格纳比罗舞台系统公司的技术主管，最近比较关注分布式计算和并行计算技术。

在这场演讲中，乔捷谈到了自己在实际的项目应用中取得初步成功的一些经验。他认为，一个分布式系统会面临可靠性、伸缩性、安全性、实时性以及异构系统之间相互集成等诸多挑战，在设计一个分布式系统时，选择合适的协议、通信方式、网络连接方式、逻辑处理模型都会对系统产生重要的影响。总体而言，.NET为构建分布式系统提供了一个良好的基础设施，快速构建一个高效稳定的分布式系统是完全可行的。同时，他对下一代C#语言在异步编程上的新特性也十分期待。

总结

本次nBazaar技术会议的召开，给中国的技术圈带来了一股全新的气象，大家抛弃固步自封的技术姿态，打破坚如壁垒的技术偏见，到技术的集市（Bazaar）中彼此交流学习。相信拥有开放心态的程序员们，会推动国内IT领域的不断进步。

■ 责任编辑：高松（gaosong@csdn.net）



RSA展会小记

■ 文 / Mars Cheng

主题令人浮想联翩

本次展会主题语是Alice和Bob的奇幻冒险，这两个经典密码学教材中的虚拟人物，成为了RSA大会的主角。但整体来看，会议主题并不是密码学。主题演讲最为核心的话题都是围绕云与安全展开的。

这实属意料之中，因为RSA的大东家EMC是云计算的业界旗帜，而云计算目前为数不多的为人诟病的地方就有安全性。RSA讨论学术的背后自然也要为公司的商业目的服务，所以EMC麾下的公司和其战略合作伙伴们一同上阵，针对此前人们对云计算的各种疑虑一一详述自己的解决方案。其实RSA此前已经倡导过自己的云计算安全解决方案，只不过有个关键性问题始终没有得到解决——其所有的解决方案都是建立在信任RSA（或者某个其他角色）的基础上的。这让业界的人们不免大失所望，因为人们希望的是一个可以不信任任何人的方案。

今年RSA自己的主题演讲便是针对各路质疑，阐明云计算的信任不但是承诺也是可以被证明的。这或多或少给了这次大会一个基石，让各路厂商展示自己对未来云计算背景下的安全理念。不过云计算的安全模型依旧是个难题，各大厂商，如微软、赛门铁克、CA等都已经在云计算上投入了大把的真金白银，不在RSA大会上一展风采，怎能善罢甘休。但各家对于云计算和虚拟化整体的安全模型还存在诸多分歧，其具体的着眼点和侧重点都有很大不同。

RSA的主题演讲中同样有着一些关于网络战争的讨论，其中有些明确指责中国的网络攻击，并用一种敌对的心态假设未来的网络战争中的敌人是中国。

传统格局未有改变：

赛门铁克就在正对大门的黄金位置，而且出入场馆长长的电梯两侧都是他的广告。本次展会赛门铁克十分下本儿，不仅做了最高级别的赞助商，而且推出了抽奖送汽车的活动。

而Intel收购，也并未让McAfee的风格和传统有任何改变，依然显现着其以反病毒为核心的解决方案供应商的传统风格。

CA作为一个反病毒厂商并不成功，逐步随着不断地并购整合变成一个集成厂商，但其展台无疑是最吸引眼球的，无论是真人虚拟场景的CS，还是高跷演员，都显示出独有的创意。

过去的技术流反病毒厂商如卡巴斯基、Sophos、NOD32等，都在随着规模成长向市场型厂商转化，这一点从他们宽敞和富有创意的站台也能看到这种趋势。

欧洲的很多二线厂商呈现被边缘化的特色，小红伞、熊猫等集体缺席，BitDefener挤在一个小站台。相比之下，倒是德国厂商的联合出展，无论从内容到形式都焕然一新。

新兴力量开始崛起

Palo alto Network在经历了多年的厚积薄发后，终于已经开始形成震撼诸侯的效应，这个以Netscreen和McAfee的资深研究人员为班底的技术型团队迅速从不足30人发展到近400人的公司，用了不到3年的时间。

很多人认为“下一代”一词显得浮滥，但Palo Alto的下一代防火墙的提法，还是为业内侧目，几乎每个来参展的“盒子厂商”，都前去刺探军情。而Palo Alto在RSA表现也不遗余力，讲座

都是CEO亲自上阵，进行了整整一天的演讲。

个性厂商怡然生存：

让我比较惊奇的是FireEye 和Damballa本次十分高调，这两个以反Botnet为主的厂商以前并不为传统力量所看好，反病毒的巨大体系资源、规范和样本积累确实令很多新兴厂商望而却步。但随着恶意代码的规模膨胀，没有任何一个厂商具有全部恶意代码的深度分析能力，而服务于特定客户，或专注于某一类分析的厂商已经开始有了崛起的空间。而在国内行业、企业级安全市场依然有鲜明的关系型色彩的背景下，深度满足特定需求的厂商在国内独立生存，几乎很难想象。

尽管我们认为IDA注定不会缺席这样的盛会，但看到时依然觉得亲切。IDA的展会人员展示了他们把ASM转化为C代码的功能。

而Norman则详细的通过大屏幕展示了其沙箱的功能。随着其技术的不断成熟，它的业务也在从单一的在线服务，走入大力推广产品级版本。

在展会上我们看到了多家，eEye等个性厂商的缺席，和ISS消隐于IBM之中，则似乎代表着扫描器的没落。

硬件环境这是个问题

本次RSA展会也是各路硬件平台供应商的秀场，从整体的ASIC解决方案、MIPS的定制平台，到专用的捕包卡、匹配加速卡等，琳琅满目。

值得关注的是Intel在会上的积极表现，尽管在PC CPU市场上高歌猛进，但作为安全设备载体这一专业领域，Intel并不开心，且不说很多安全设备的低端型号纷纷倒向ARM，另外一些高端型号也有人押宝。就算在x86架构上，也有更多人选择了GPU。

因此Intel本次会议有相关业务线的高管到场，早上8点开始训话，之后Intel展台上所有人一起扑向其他厂商的展台，宣讲Intel最新安全平台，号称能达到8~10G的性能。其千核CPU也在近期发布。网络安全设备载体是一个小市场，但其性能却关乎着芯片老大的专业形象，不得不重视。

中国面孔

在展会前一天的Party上，我们吃到了煎饺和春卷，这时我们才注意到，周围有很多中国面孔。

本次有十家中国企业来到了RSA展会，华赛、绿盟、山石三家独立出展，而启明星辰、天融信、网御神州、安天等在中关村管委会的组织下，组成了一个很大的联合展台。中关村展区位于近乎会场中心的地带，周围都是大公司展区。RSA的主席也专程来拜访。中关村此次活动组织得可谓非常给力，从现场组织到后方补给面面俱到。

希望明年，我们能在RSA看到更多的国内同行。

作者简介 | Mars Cheng

悉尼大学计算机专业硕士，安天实验室产品与项目管理中心高级产品经理。

■ 责任编辑：高松（gaosong@csdn.net）

明确规划，乐于沟通

——微软MVP蒙洋专访

■ 记者 / 杨东杰

蒙洋是北京中芯优电信息技术有限公司的CEO，兼任北京航空航天大学软件学院讲师，主讲《嵌入式应用》课程。他善于教学并乐于分享，发表过多部技术著作和学术论文。在2010年，蒙洋当选为微软嵌入式最有价值专家（MVP）。

从普通的技术人员一步步成长为创业者，蒙洋认为明确的职业规划和高效的沟通技能对技术工作者最为重要。

记者：根据您与技术开发学员的交流经验，他们比较集中的困惑是哪些？

蒙洋：首先，很多学员不知道如何选研究方向。例如，嵌入式学员经常会问该学Linux还是WinCE。我个人的建议是学生应该先精通一种，另外一种也很容易学会，其中很多知识是相通的。

其次，有些学员迷信学习捷径。我认为没有一蹴而就的快速学习方法。对开发者来说，唯一有效的方法就是动手实践，只有潜心钻研，虚心请教，精益求精，才能真正掌握技术的精髓。

记者：您如何看待微软WinCE技术在国内嵌入式市场的发展情况？

蒙洋：目前国内嵌入式开发技术正呈现百家争鸣的态势，随着物联网和云计算等新兴技术和服务的发展，嵌入式技术正经历重新洗牌。

在手机市场，Android系统已占据领先优势，但在电子开发板、工业人机界面等行业领域，WinCE还保持着一定的优势，最新的Windows Embedded Compact 7系统功能强大，而微软对嵌入式市场也格外重视。

对于开发者来说，虽然WinCE有着不开源、需要授权费用等缺点，但相对其它技术性能稳定、开发周期短、开发难度低。我在大学期间就考过微软嵌入式方向认证工程师，之后一直从事基于WinCE嵌入式开发。

此外，许多政府部门和大型企业都在大力发展物联网，这必将带动嵌入式行业的快速发展。

基于行业发展背景和信息技术专长，我们在2010年成立了北京中芯优电信息技术有限公司，产品方向主要是基于Windows Embedded Compact 7的智能家居，以及物联网实验系统。

记者：请谈谈您的技术经历和成长感悟。

蒙洋：我是电子工程科班出身，在大学期间主要有两个收获，一是掌握了基础的编程技术，并通过了程序员认证考试。二是明确了自己的职业发展方向，首先从技术人员做起，然后创立自己的信息技术公司。

多年来，我一直坚持践行这份职业规划。

大学毕业后，我的第一份工作是普通的技术人员，虽然辛苦但收获不少，在此期间我积累了许多软硬件研发技术实践经验。我曾在清华紫光微电子担任研发工程师，不仅增进了技术水平，而且对企业管理和培训有了更深入的了解。

蒙洋告诫初学者不要迷信学习中的捷径

此外，我在嵌入式培训行业也逐渐精进。2010年我被评选为微软嵌入式技术MVP，经常受邀参加微软嵌入式技术的讲座和学术会议，并作为微软TechED 2010主讲老师完成了Windows Embedded系列动手实验室课程，与同行的技术高手建立了良好的联系。

在北航读研期间，我结识了一群热爱技术的年轻人，我们组成了创业团队，自主研发嵌入式产品并投入市场。

百度CEO李彦宏有一句话很有道理，他认为如何使自己的产品和更多的人发生关系是企业做大的关键因素。我选择智能家居产品就是认为智能家居会逐渐走入老百姓的家庭生活，更好地服务于大众。

■ 责任编辑：高松（gaosong@csdn.net）

妇女节特别专题

铿锵玫瑰更从容

女程序员自我成就三步曲

潘正磊

微软Visual Studio商业软件部总经理、微软亚太研发集团服务器与开发工具事业部中国团队的高层领导团队成员之一。

勇气和自信至关重要

在男性占据主导的软件行业中，女程序员往往更容易引人注意。如何让自己在众多男性同行中脱颖而出，并赢得他们的尊重？勇气和自信至关重要。

对于在男人堆中工作的女程序员们，需要有更多的勇气，并积极主动敢于进取。与同事们讨论问题时，要乐于表达自己的想法，它并不一定成熟，甚至可能遭到反驳，但只有提出来，同事们才能了解你的见解，在讨论中共同完善这个新的创意或解决方案。当面临挑战，要敢于主动接受，即使失败了，总结失败教训的过程也是自我提高的过程。

大家往往觉得女性在职业生涯发展上存在天花板。其实，所谓的“职业生涯天花板”很大程度上产生于个人主观意识。一旦认为有“天花板”高高在上，你也就主动放弃了向上发展的所有机会；反过来即便有“天花板”，但相信自身实力，勇于尝试不同的方式打破“天花板”，你就有机会走得更高。

不断提升能力，发挥自身优势

经历几年编程工作后，程序员们大都面临着技术能力要迅速提升所带来的压力与挑战。技术能力的提升是一个循序渐进的过程，不可能一蹴而就。每年你都应该选定一、两个目标，而且一定是那种“需要踮起脚尖”才可以够到的目标，然后认真完成。在微软工作的18年来，我就是这样成长起来的。

除了技术能力外，如何在工作中尽可能发挥自己的优势，对你今后的职业发展方向也有很大的影响。记得担任开发经理之前，鉴于我比较强的技术能力，当时的老板建议我向架构师发展，建议我多观察身边的几位架构师，和他们多聊聊。经过一段时间的观察和比较，相较于架构师们对技术的狂热，我觉得自己更乐于并擅长发现别人适合做什么样的工作，而且我很喜欢帮助别人成长，这些会让我更快乐，由此我开始了我的技术管理生涯。所以每位程序员都应该仔细想想自己在团队里发挥着什么样的作用，与别人差异化的优势是什么，选择符合自己的职业发展方向，从而发挥自己的优势。推荐大家一本书——《Now, Discover Your Strengths》，中文翻译为《现在，发现你的优势》。我觉得很值得一读，可以帮助你更好地认识自己，最大限度地发挥那些有助于你成功的才能。

刚柔并济，助力女性技术管理之路

领导才能、富有远见、合理调配资源是每位管理者应具备的基本素质。对于计划转型做管理的女程序员，特别需要注意展现自己的观点或看法，不断在工作中和团队里提升自己的影响力。在一个男性居多的工作环境中，采用怎么样的方式去扩大影响力，是女程序员转型过程中一个非常值得花时间思考和实践的问题。只有获得同事和下属的认可与尊重，你才可能带领团队齐心协力地完成既定目标。

作为技术管理者，女性无疑比男性面临着更多困难。首先可以学习、借鉴的女技术管理者榜样并不多，周围可以交流经验的更少。其次，较之男性管理者多样的管理风格，女性可选择的则少之又少。过于强硬或过于温柔的管理风格，都只能引人侧目，无法建立管理者应有的人格魅力和威信； 只有“刚柔并济”，拿捏好软硬尺度才能成就一个优秀的女性管理者。

女性技术管理者更能精准把握人际关系

李严冰

VMware中国研发中心总经理、清华大学电子工程学士、美国普林斯顿大学电机工程博士。

在VMware中国研发中心超过200名全职员工中，大概有20%的女性员工，具体到核心技术团队的女性编程人员，则只有10%。“女性不适合做程序员”这种传统观念在很大程度上抑制了女性进入软件行业的积极性。但从女性工程师整体表现及公司对她们的各种评估上看，她们在能力上与男员工不相上下。对于一个公司来说，凡通过招聘标准入职公司的员工，皆是达到公司要求的，而不存在明显的男女差别。

在我看来，只要对技术有足够的热情，把编程当做自己的兴趣爱好去追求；勇于探索新技术，不仅仅拘泥于工作所接触到的东西；具有很好的分析能力、团队协作能力的程序员均可以成为优秀的程序员。虽然女性在思维方式、生理、心理上与男性有所不同，但只要热衷于自己喜爱的，女程序员同样可以干得很出色。第一位女程序员Ada Lovelace就是一个很好的例子。

由技术转型做管理，对于IT技术人员来说是一个比较常见的发展道路。由于对人与人之间的关系把握的更精准，女性更容易进入管理这一角色。诸多实践证明，女性也容易胜任管理者这个职位。据国外权威数据统计，高层管理层中有女性代表的公司比缺少女性代表的公司在商业业绩上要高出35%。这一数字说明，增加女性员工对提高公司的运营与营收有着战略性的意义。

然而作为一名女性管理者，尤其当男性员工占绝大部分时，面临的最大挑战就是在技术上是否具有可信度，是否能让团队信服你、尊重你。“尊重你”并不是说尊敬你是一个管理者，而是尊敬你是一位懂技术的人。你的技术不一定是最好的，但必须要懂，可以说技术是你做好管理工作非常重要的基础。然而技术又仅仅是非常小的一个层面，对于管理者来说，衡量你工作能力的是你领导团队所创造出来的成果，即领导力和执行力才是最最重要的。

从这个角度来说，管理其实是一个比编程工作复杂很多的工作。对于那些想成为管理者的女性IT技术人员来说，首先要衡量一下自己是否喜爱管理工作，相比于与机器打交道是否更乐于与人交流、与人互动，不要以为管理工作薪资高、被人尊重而去选择。

家庭与事业，择其重要者而为之

贾茵

解决方案专家，曾获微软Dynamics CRM产品认证专家、微软CRM MVP称号。

“妈妈快看，车车！”儿子高高举着刚用拼装玩具拼好的汽车，兴高采烈地跑过来向我展示他的新成果。“宝宝真棒，真好看！”我把视线从正在编写的方案标书中移到儿子手中的汽车上，狠狠的夸奖了他，其实是为了下面这句话：“宝宝乖，妈妈在工作，去客厅玩好吗？妈妈忙完了就跟宝宝一起玩。”其实我知道这是一个“空头支票”。说完起身关上房门，把儿子挡在门外，关门的一瞬间，儿子眼里充满了失望，但是很懂事的走开自己玩去了。不到2岁的他，已经很懂得在我工作的时候不去打扰，而我，亏欠他的太多。

混在IT业近10年之久，角色从一个小小的

Web程序员变成IT主管，又成为售前工程师；而在家里，角色从单一的女儿，增加了妻子、儿媳、母亲……这些轨迹，看似男版女版都一样，实际上女性要承担的可能更多。在一次与男同事聊天的时候，他们表示下班后宁愿在公司加班，因为回家后事情太多，既要做家务又要照看孩子……想起刚生完孩子那段时间，因为每天早出晚归正好与孩子活动时间错过，以至于孩子对我越来越陌生，后来便毅然辞职在家陪孩子，直到他对母子关系的认知达到稳定的年龄阶段，我才又重新找工作。

对于天天都需要学习了解新知识的IT从业者来说，每天8小时之外的学习或处理剩余工作的时间，大部分都安排在晚上孩子睡觉之后，于是熬夜成了家常便饭。再加上到点喂奶、换尿布、盖被子……可以说从孩子出生到现在，我几乎没有睡过一个整觉。

不只是孩子，其实女IT人从一开始，在家庭问题上就面临着比男IT人更多的问题，例如择偶。男性择偶选择范围非常广，而IT业内的女性很多只在同业中寻求自己的另一半。因为IT业内的女性多被视为缺少女性的柔美、过于理性、好强、过于有主见等等，很多男性对之不感兴趣。

但我想，既然选择了，就接受吧。平衡好工作和家庭的关系，同时得到家人更多的理解和关爱，也是对自己的巨大挑战，例如时间安排的能力。两者之间的平衡处理不好之时，停下来想一想哪个更重要一些，不失为一个很好的方法。实在无法兼顾的时候，则可以考虑暂时放弃其中的一方面。多看看同业其他女性的传记，也是一种非常好的借鉴方式，这里推荐看看前惠普CEO卡莉•菲奥莉娜的自传《勇敢抉择》。

不要寻求歧视——写给求职中的女程序员们

张大志

zhaopinpro创始人，《程序员羊皮卷》等图书作者、HR咨询顾问。

某次针对211高校计算机系同学的就业讲座后，一位女同学提问：“研发领域女生工作机会比较少。如果我像男同学般要求自己，能有更多的机会吗？能不被歧视吗？”我的观点是：不要主动寻求歧视，我们总能找到自己的位置。

研发领域给女性提供的职位少也许是个客观现象，有调查表明大学里相关专业的男女程序员比例与实际工作中的比例相当，说明大多数公司并不存在性别歧视的问题，更可能的原因是：学习计算机专业的女生本来就不多，相应的职位自然不会太多。

据说有些公司在招聘时会问“您最近有结婚打算吗？”、“是否有生孩子计划呢？”此类问题，明显暗示如果你打算结婚生子，那我们公司怕是不会考虑你了。对此我倒是建议女性不要委屈自己、告诉对方“我最近没这个打算”继而去迎合这种歧视。干脆不要考虑这样的企业。为人母是大多数女性人生非常重要的一部分，没必要因为工作而完全放弃自己的生活权利。更有些不靠谱的企业内部规定“女程序员不要”，遇到此种情况我们应该马上转身离开，不要主动寻求歧视。有如此规定的公司，即使是大企业、大公司也不必考虑。因为他们已经定下不要女性成为程序员，即使入职，以后发展机会也同样渺茫，身为女性根本没必要为证明什么而非要加盟这样的公司。

爆个内幕，我每次给500强企业提供面试服务时，发现他们针对研究性岗位，一般会提出如下要求：“女生只要学习成绩别太差、人够聪明，务必让她们通过初试。”某知名企业CTO解释说：“你想想天天上班面对一帮男的，能有个女生调剂下气氛也是好的，很有助于提高工作效率啊！困难在于我们招不到女生，一来是学计算机的不多，二来很多企业在抢。”可见真正优秀的公司不但不存在性别歧视，甚至有偏袒女性的倾向，所以不必在有歧视地方浪费太多时间。

回到开头的问题，我当年的回答是：“您本来是女生，为什么要像男人一样呢？发挥自己的优势与特长，找出与别人的不同；找到自己的兴趣，不断努力，相信您很快能找到真正属于自己的位置。”

当今的社会机制对有能力、愿付出的人会有更多机会，女生也同样。所以女程序员们，一定要勇敢起来，寻找适合自己的环境和发展机会！

计算机如此重要，不能只留给男人做

——计算机界五位伟大女性介绍

2011年三八妇女节已经到来，在软件业这个由男性称霸的世界里，其实还有着很多的巾帼英雄，本文我们推荐五位计算机界的伟大女性，她们的成就、她们的努力，将带给我们前进的动力。

Ada Lovelace

世界上第一位程序设计师

Augusta Ada King原名Augusta Ada Byron（1815年12月10日—1852年11月27日）。她是著名英国诗人Byron之女。

在1842年与1843年其间，Ada花了9个月的时间翻译意大利数学家Luigi Menabrea对Babbage最新的分析机概论所留下的备忘录。在这部译文里，她附加了许多注记，内容详细说明了用分析机进行伯努利数的运算方法，被后人认为是世界上第一个电脑程序。

Ada的文章创造出许多Babbage也未曾提到的新构想，比如Ada曾经预言道：“这个机器未来可以用来排版、编曲或是各种更复杂的用途。”

1852年，Ada为了治疗子宫颈癌，却因此死于失血过多，享年36岁。具有讽刺意义的是，她与她父亲Byron死于相同年龄，一样死于治疗中的失血过多。Ada的生命是短暂的，但她对计算机的预见却超前了整整一个世纪。

在1980年12月10日，美国国防部制作了一个新的计算机编程语言—Ada。而美国国防部标准局为了记念Ada，以她的生日设立了一个编号MIL-STD-1815。在微软的产品里也可以找到Ada的全息图标签。另外英国计算机公会每年都颁发以Ada为名的奖项。

Grace Hopper

计算机软件的第一夫人

被誉为计算机软件第一夫人的Grace Hopper于1906年12月9日出生在纽约市的一个海军世家。Grace Hopper是杰出的女数学家和计算机语言领域的带头人。

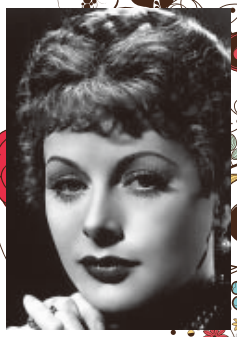
1949年Hopper加盟由第一台电子计算机ENIAC发明人埃克特和莫齐利开办的电脑公司，为第一台存储程序的商业电子计算机UNIAC编写软件。1952年，她开发了世界上第一个将高级符号语言转变为机器语言的编译器A—0，第二年她又开发出第一个处理数据计算的编译器A—2以及第一个自动翻译英语的数据处理语言。

之后她又以Flow-Matic为基础开发了COBOL语言。COBOL被称为第一批高级程序设计语言之一，并广泛用于大型机和小型机电脑的高级商业程序设计。同时Hopper又率先实现了第一个COBOL编译器，因此被誉为COBOL之母。据20世纪80年代初的统计，当时全美国有80%的程序由COBOL语言编写而成，此语言对计算机应用发展有着很大的推动作用。

Hopper致力发展程序设计技术，同时还培养了大批的程序设计人员。Hopper自己曾说：“与其说我的最大贡献是发展了程序设计技术，不如说我培养了大批程序设计人才”。

在Hopper传奇的一生中，她赢得了无数荣誉和奖励，她先后被40多所大学授予荣誉博士学位。

1971年为了纪念现代数字计算机诞生25周年，美国计算机学会特别设立了“Grace Hopper”奖，颁发给当年最优秀30岁以下的青年计算机工作者；1980年Hopper获得国际IEEE组织颁发的首届计算机先驱奖；1991年，布什总统在白宫授予Hopper“全美技术奖”，这也是至今美国女性唯一获此殊荣的人；1994年Hopper被追授为“美国女名人”，进入“全国女名人堂”。



Hedy Lamarr

信息技术史上最传奇的女性

Hedy Lamarr（1913年11月9日—2000年1月19日）美国好莱坞默片时期著名女演员，生于奥地利一个富裕的犹太人银行家庭。她曾与克拉克·盖博等顶级男星合作，主演了多部热门影片。19岁时在电影《神魂颠倒》中裸泳成为世界电影史上第一位裸体出镜的女演员。一生曾有过八次婚姻。

她曾被认为是全欧最美的女人，并因美貌盖过了演技而被人称为花瓶。有意思的是，她其实数学和通信功底很深，是现代无线通信的核心专利跳频技术的第一发明者，CDMA、WiFi等技术都以此为基础。美国电话局主席安东尼•罗德（Anthony Loder）对她的评价所说：“虽然Hedy Lamarr已经被大家遗忘了，但她所做出的一切仍然影响着一代又一代的人。”

2005年，德语国家举行了第一届发明者节，纪念她的92岁诞辰。另外，大家都很熟悉的CorelDraw 9软件封面上的完美面容也属于Hedy。所有的这一切，仿佛在印证她的一句妙语：“电影往往限于某一地区和时代，而技术是永恒的”。

Frances Allen

第一位获得图灵奖的女性

Frances Allen，著名计算机科学家，作为一名编译器优化领域的先驱，她的成就主要包括编译器的基本原理、代码优化和并行编译等。

1954年，Allen毕业于美国纽约州立大学奥尔巴尼分校，并获得学士学位。她于1980年代早期创立了并行翻译（Parallel TRANslation，PTRAN）研究组，致力于研究并行计算机的编译问题。该小组的工作在编译器的并行化方面处于世界领先的位置。她在这些项目中的工作促成了许多目前广泛应用于商业编译器中的程序优化算法和技术。

Allen在科学的道路上探索奇妙之旅，获得荣誉无数。Allen在IBM业界有着广泛的影响，1989年Allen当选为IBM院士，这是IBM历史上第一个女性获得此殊荣；1995 年，她被任命为IBM技术研究院院长；1997 年被选入WITI名人堂；2000年IBM设立了以她的名字命名的“Frances E. Allen科技女性导师奖”。2007年2月，作为美国国家工程学院院士、美国计算机学会会士，获得过AWC颁发的Augusta Ada Lovelace奖的Allen因“她对于优化编译器技术的理论和实践做出的先驱性贡献，这些技术为现代优化编译器和自动并行执行打下了基础”而成为第一位获得图灵奖的女性。图灵奖评委会主席Ruzena Bajcsy说：“她的研究几乎影响了计算机科学发展的整个历程，使我们今天在商业和科技领域内使用的许多计算技术成为可能。她此次获奖进一步证明成就与性别无关。”

Barbara Liskov

CLU与Argus语言发明人

计算机界有“计算机界诺贝尔奖”之称的图灵奖曾由男性垄断了40年，而在2008年6月这一局面再次被打破。Barbara Liskov被授予2008年度图灵奖得主，以表彰她对编程语言和系统设计方面所做出的实践与理论基础，尤其是数据抽象、容错和分布式计算方面的贡献。她也是第二位获得此奖项的女性科学家。

Barbara Liskov，本名Barbara Jane Huberman，1939年生于加利福尼亚。1961年在加州大学伯克利分校获得数学学士学位。在20世纪60年代，计算机科学这门新兴职业对女性来说还相当寒冷。Liskov在申请研究生、找工作过程中屡次碰壁。但这并没有击垮她，她认为“发生的不公平的事情，并不与我直接相关，我想也许正是这种态度，使我已经适应这些年来此类处境。”

Liskov生平最重要的科研成果是她为推动数据抽象使用所做的巨大贡献。她在此领域的创新使得软件更易于编写、修改和维护，极大地提高了计算机软件的可靠性、安全性和易用性。Liskov从实际项目中提炼出来的数据抽象思想，已经成为软件工程的重要精髓。

20世纪70年代早期，Liskov发明了两种计算机语言：CLU（一种支持数据抽象的面向对象编程语言）和Argus（一种分布式程序实现的高级语言）。这些研究成果成为现代编程语言的基础，支撑起整个现代应用软件行业，对每一种主流汇编语言产生了深远的影响，如C++、Java、Python、Ruby、C#等。她与亚裔女科学家周以真一起提出的Liskov替代原则，是程序设计中另一个广泛应用的成就。这个原则已成为面向对象最重要的原则之一。

搜索引擎王者应具备的六大特质

腾讯搜索技术研发中心总经理孙良认为，大学里的基础课程很重要，例如数据结构和算法，这些都是日后从事搜索引擎技术工作的基础。

孙良 腾讯搜索技术研发中心总经理

记者：请简要介绍一下您的求学和从业经历。

孙亮：我是浙江大学信息学院计算机系硕士毕业。在海量信息检索和搜索引擎技术领域打拼超过15年。2006年加入腾讯，受命负责组建腾讯搜索业务核心技术团队。加盟腾讯前，曾在中搜公司信息产品事业部任研发总监，主持研发了包括“随e搜”在内的多款搜索产品。

记者：您对搜索引擎目前的发展现状怎么看？

孙良：首先，现在的搜索只能通过简单的搜索框来完成搜索，给用户表达信息的地方比较少，因此很难确切地告诉搜索引擎：用户到底想要什么，因为汉字往往存在歧义，如搜索“苹果”时，结果页既有水果，也有手机。其次，传统搜索方式从打开浏览器输入检索词，到甄选搜索过程，搜索结果跟提交搜索前用户在做什么没有直接关系。而腾讯搜搜提倡的“情境搜索”，与传统的个性化搜索区别在于，不仅强调对用户的兴趣建模和行为分析，还希望结合用户具体使用搜索时的场景，并针对个人长期和短期搜索兴趣之间做好平衡，以返回最确切的结果。

记者：未来的优秀搜索引擎需要具备哪些特质？

孙良：作为一种集成技术的综合应用，需要具备：第一，产品的服务模式及对用户群体的针对性。第二，海量的高度黏性的特色用户数据资源以及对这些数据进行挖掘分析的能力。第三，平台的性能和架构。第四，对各种各样搜索关键技术的集成能力。第五，移动领域的资源和实力。第六，可持续增长的商业模式。

记者：实现“情境搜索”的主要技术难点在哪？如何让这一理念逐步落地？

孙良：除了对海量数据的挖掘和分析匹配、跨界产品设计和开发相关技术，还有几个特点和趋势值得关注：首先是交互性，例如人与系统、系统之间、数据与系统交互。推荐、协同处理、问答都可算作交互的一部分。如今基于这类交互的计算模型还不清晰，往往在明确的场景环境下，可以通过确切的技术来实现，但是在场景瞬息万变的情形下，如何能给出快速精确、有针对性的搜索结果，相关的理论还需要进一步研究。其次，UGC（用户产生内容）的正向滚动。此外，搜索的社区性，或者叫做局部性，更多体现在多种属性、多重因素的复杂异构搜索结果的关联性模型。第四，基于数据安全、信任和用户隐私保护的问题。

至于落地方式，一种情况是通过PC，将结合腾讯各个产品间的搜索。另一种情况是基于移动终端的产品，与LBS（应用）相类似，目前主要是结合腾讯的手机地图应用。

记者：对于有志做搜索引擎的年轻人，您有哪些建议？

孙良：如果是做相关的技术工作，有些知识是必须掌握的。首先是数据结构的算法，这是很基础的一点。因为现在的搜索跟二十年前的单机版、小型垂直检索系统相比，复杂度上已经不可同日而语了，需要考虑更多信息压缩、倒排索引的组织等等。其次是大规模（海量索引数据）的集群和并行计算技术，像搜搜处理的数据索引已经过百亿。对索引的处理要涉及到很多排重去噪、垃圾索引识别和反作弊技术等，在这个过程中，要有大量的服务器节点去参与运算，因此并行计算的技术也要去了解。有时甚至要了解PC服务器的体系架构是什么样的，如何利用一些特别的指令来加速对索引的处理、压缩、解压缩和跨节点之间海量数据的通讯机制等等。除此，还应关注对搜索引擎质量的评价手段。

当然，上面是些总的技术点，一个人不可能把这些技术全部学完，可以根据个人的兴趣和定位不同，有自己的侧重。（记者/付江）

尊重人才能“事半功倍”

网票网技术总监周华林认为，不好管理的团队成员往往有“专长”，所以要有耐心。只有把团队成员当作人才来尊重，才能充分发挥大家的积极性，实现事半功倍。

周华林 网票网技术总监

记者：您最初接触计算机是什么时候？如何开始IT职业生涯的？

周华林：记得读高一时，学校组织各类兴趣小组，我选择了当时的“高科技”——计算机。这是我首次与计算机结缘。由于大家都没有接触过计算机，所以老师不得不从基本操作开始教起，然后才教如何编写简单的程序。我就是从这个时候开始迷上了编程。最终走上IT职业生涯，很大程度是因为一路上得到了各个计算机老师们的耐心帮助，其中最想感谢的是我大学时代的导师——赵恒永教授和徐南山老师，他们经常到实验室来指导我们，纠正方法上的错误。那时我在校期间参加了不少项目，真正得到了锻炼，并积累了项目经验，这对我后来的职业生涯起到了很大的帮助。

记者：您为什么加入网票网？目前的工作重点是什么？

周华林：主要是“电子影票”吸引了我。网票提出让“电子影票”和互联网及手机应用结合起来，我认为这是一种非常新颖且符合人们生活需求的运营模式。实现网上实时选座，提前购买电影票，而且在影院现场自助出票，在当时国内还没有一家公司能这样做。加上国内越来越火爆的电影市场，也是我看好“电子影票”未来发展的原因。

网票技术部门的工作重点主要是进一步做好网票的运营支撑。用户购票高峰期的数据处理能力，是目前运营系统存在的最大问题。首先，由于“电子影票”从用户查询影院、选座支付到现场实时出票这个过程涉及到多个设备和模块通信问题，一旦一个模块出现瓶颈或出现故障，会导致整个流程都变慢。其次，目前国内大多数影院的网络环境较差，电子订单的流程往往会出现故障。此外，接入的影院越来越多，海量数据处理也使我们面临全新挑战。

记者：与竞争对手相比，网票的最主要优势是什么？

周华林：网票独有的QR二维码认证技术是最主要的优势。它由我们公司率先研发，目前已经相当成熟。这项技术来源于日本DENSO公司在1994年制定的二维条形码标准，其本身具有非常多的优良特性，但是要把它应用到“电子影票”上，必须要做二次开发。“电子影票”是把电影票的相关信息都以一定的编码方式存储在QR二维码内，而二维码的读取过程就是照相和识别的过程，其读取设备首先照相获得二维码图片，然后通过图像处理技术使图片更加清晰，最后进行识别。此外，相对同类公司，网票还具有“电子影票”、电子支付以及运营综合方面的优势。

记者：作为技术总监，您平时是如何管理团队的？

周华林：首先，我会定期召开会议制定技术工作方向、了解各个项目执行情况，指导大家解决问题。其次，我会按照考核指标，每月对团队成员进行考评。我认为，技术人员是企业的核心，要用合理的激励机制去留住这些人才。所以在管理上要“以人为本”，发挥“民主”机制，调动团队的积极性。对于不好管理的员工，要有耐心，因为这样的人往往都有一定“特长”。总之，只有真正把员工当作人才来管理，尊重人才，才能够最大地发挥员工积极性，才能够“事半功倍”。（记者/张雪峰）

2011开放平台之征

■ 策划 / 本刊编辑部

本期封面报道，将介绍开放平台的种种议题。

我们请两家微博平台同台论道，试图从中得出不同的开放思路和经营理念。微博在近两年呈井喷之势，各大门户网站都推出了自己的产品，在国内的进化速度甚至超过了国外的原型。但任何一家若要持续繁荣，“开放”与他人是必然的选择。

我们论述两家SNS网站平台，探讨形成开放生态系统的可能。若说SNS位于Web 2.0的核心并不为过，如何让单一相度的社交网络变得立体，通过整合Social、Location、Mobile，以“开放”的姿态，打破日趋板结的自有平台，才能引入“源头活水”，生生不息。

我们还将开发者平台和电子商务平台并置，展示开放的多样性与多重性。它们都使得创业成本极大降低，对个人参与的“开放”，促进社会发展的同时，也让自身不断地壮大。

即使是两两相较，“开放平台之征”也不完全是彼此的“征战”，还意味着大家都已经不可避免地一起踏上开放的“征途”。

整合产业链是关键

——关于开放平台的一些思考

■ 记者 / 常政

开放平台（Open Platform），自2007年因Facebook取得瞩目的成功以来，一直刺激着国内互联网界的神经。2008年起，天涯、康盛创想、51、人人网等陆续开放了自己的API。但值得注意的是，自进入2010年后，这股开放平台潮突然“加速”了：1月，淘宝推出应用商店淘宝箱；5月，此前信誓旦旦1年内不开放的开心网突然妥协；7月，新浪推出微博开放平台；9月，腾讯推出社区开放平台；而百度则提出了“框计算”……与此相关的，长期处于弱势地位的开发者群体也突然成为香饽饽，成为各大平台商热捧的对象。开放平台，绝对是2010年中国互联网最火的关键词。这股火为什么突然蹿得那么旺？

互联网竞争的全新高度

开放平台是什么？根据维基百科的定义：“开放平台指在软件业和网络中，软件系统通过公开其应用程序编程接口（API）或函数（function）来使外部的程序可以增加该软件系统的功能或使用该软件系统的资源，而不需要更改该软件系统的源代码。”

非技术背景的读者可能心生疑惑，这个看似有些晦涩的以Open API 为基础的开放平台缘何突然成为各财经媒体封面报道的明星、互联网淘金者眼中的宠儿？

其实从开放平台更加广义的内涵看，它对于我们并不陌生，在我们的日常生活里随处可见。比如超市、菜场，它们提供场地、物业以供商贩们入驻销售；甚至我们电脑里可以支持各种应用软件的Windows，其实也是一种开放平台……所谓开放平台，本质是配备流通渠道的资源汇集地。而一进入互联网世界里，物理实体形式的资源转变成了“二进制数据”，诸如菜农的一系列具体经营行为则相应地被抽象成了一个Open API 函数。

就这样，商机诞生了，比如通过一软件应用商城的API，开发者可以写个入口程序，既可把自己的作品导入商城销售，也可以将商城的销售商品导入自己的应用，帮助商品寻找更多的新买家。据统计，Facebook每天有50亿次API应用请求，占据它全部流量的75%——从这我们不难简单一窥整个Open API 体系蕴藏的市场前景。

同时，和2010年同样风光一时的关键字：“团购”、“地理位置应用（LBS）”所截然不同，开放平台不是单纯的商业模式，而是一种资源的调配方式，势必会影响互联网的产业格局。而随着各开放平台的争相齐放，意味着整个业界已经达成“共识”：现在中国的互联网竞争已经进入了一个全新的高度——整合资源为核心的产业链竞争。

开放平台的类别

按照开放的内容属性，笔者将开放平台分为三种形式：

- **开放数据**：主要提供平台数据的检索调用、以及相关的营销服务。目前比较热的有用户关系数据、地理位置信息数据、产品数据等。这是当前国内开放平台的主流。
- **开放服务**：开放的是某种功能服务，如亚马逊提供计算资源的云计算服务。这类云计算服务，目前国内在政府的支持下，正在如火如荼的前期建设阶段，相信几年后会兴盛起来。
- **综合系统服务**：指通过Open API 所调用

的数据与服务延伸到了实体世界，目前主要应用在电子商务企业。比如2011年2月，当当网透露将启动的“物流开放平台”，可为电子商务企业提供商品储存、分拣、包装及全国1200多个城市的货到付款式配送服务。这种模式尽管目前方兴未艾，但随着物联网时代的到来，数字世界与实体世界的整合更加密切，无疑将是未来开放平台的主流。

开放平台的经济学原理

从经济学的角度出发，开放平台的内在驱动力在哪里呢？答案是长尾理论。福布斯去年所发表的《什么刺激了API淘金潮？》指出：“创建和应用API的潮流是靠长尾经济驱动的。生产工具本身基本是免费的。开发者能够为各种特定的需求开发应用程序。即便每个应用程序都不是非常流行，但当几千个应用的流量累积在一起，就十分巨大了。”

爱立信研究员、开放平台实验室联合创始人刘青焱在其文《开放平台的经济模型》中，绘制了基于长尾理论的开放平台经济模型（如图1）。

由图1可看出，平台提供者只要满足了长尾的用户需求，减缓了成本增长，便可创造额外的利润（详细论证参阅：http://oplatform.org/archives/73）。而根据姜奇平（《互联网周刊》主编）先生在其著作《长尾战略》的界定，长尾经济属于“数字范围经济”，其特征是“初始固定资本投入高，边际投入低，边际收益递增，边际成本递减”。同时对于第三方开发者来说，他的利益取决于平台的用户群和分成比例。所以一旦平台不能采取有效措施吸引开发者，它的成本将增高，造成两败俱伤的结果。至由此可见，足够大的用户群、合理的利益分配体系、严格的制度保障将是开放平台生态体系健康运行的关键。作为平台提供者，只有以一种真正开放的心态、力促产业链整体共赢才能获得长久的发展空间。

开放平台的中国特色

在理想化的经济模型里，彻底“开放、分享”更能带来产业链的共赢，但或许是中国互联网长久以来的“丛林环境”，知识产权、行业规范的不完善，导致中国的开放平台潮流与西方相比，总显得“南橘北枳”、呈现某种异数。中国的平台商首先考虑的往往不是打造健康的产业链，而是借“开放、联盟”之名，大肆圈地、扩大声势，以至整个业界反倒出现一番征战杀伐的气象。盘点下来，有如下规律或特征。

犹抱琵琶半遮面。开放性不够，是目前

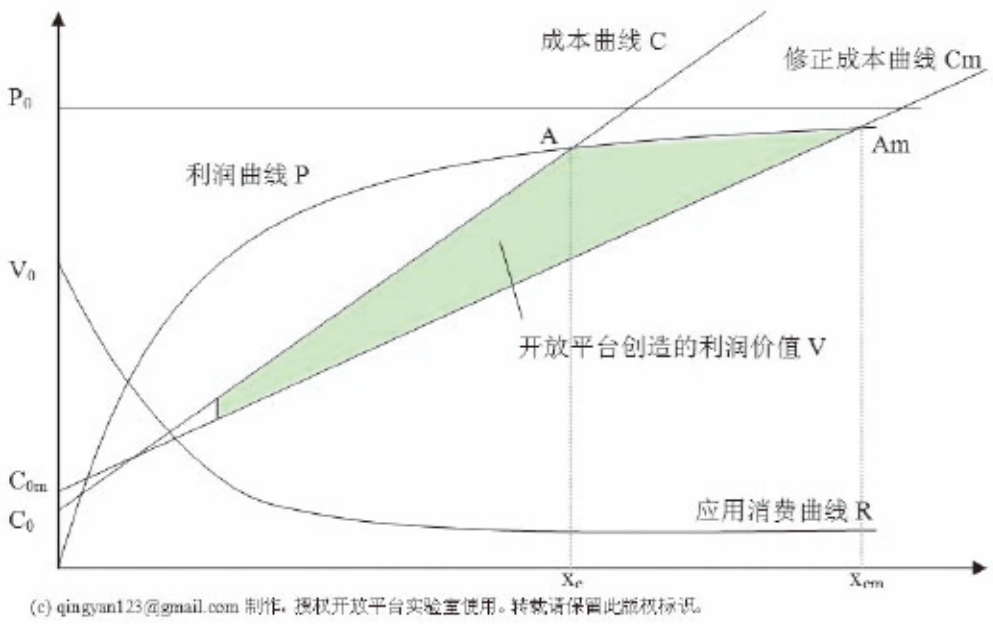


图1　开放平台经济模型

表1　国内主要开放平台网址

新浪	新浪微博开放平台 http://open.t.sina.com.cn/
千橡	人人网开放平台http://dev.renren.com/
开心网	开心网开放平台http://www.kaixin001.com/platform/
阿里巴巴	淘宝开放平台http://open.taobao.com/
百度	百度知道开放平台 http://zhidao.baidu.com/s/open/
	百度开放平台http://open.baidu.com
	百度应用开放平台 http://app.baidu.com/developer
	百度移动开放平台http://open.shouji.baidu.com/
盛大	盛大开放平台http://open.sdo.com/
	盛大开发者平台 http://dev.sdo.com/
腾讯	腾讯社区开放平台http://opensns.qq.com/
	财付通开放平台http://open.tenpay.com
	拍拍开放平台http://pop.paipai.com
	搜搜开放平台http://open.soso.com/
	腾讯微博应用开放平台 http://open.t.qq.com/
	Manyou开放平台http://www.manyou.com/www/
网易	网易微博开放平台　http://open.t.163.com
	网易开发者平台http://apps.163.com/
天涯	天涯开放平台http://sandbox.tianya.cn/
优酷	优酷开放平台http://dev.youku.com/
搜狐	白社会开放平台http://wiki.bai.sohu.com/
	搜狐博客开放平台 http://ow.blog.sohu.com/
51	51开放平台 http://developers.51.com/
豆瓣	豆瓣API http://www.douban.com/service/apidoc/

各大开放平台最为业界所诟病的。开放的名义下，附加的保守条款让第三方开发者们眉头紧皱。例如人人网的开发协议“人人网确认对开发者提供的插件应用组件，免费供人人网用户使用，但以后是否收取费用，收费的时间、费用额度，均由人人网决定”；例如豆瓣网官方条款规定“不得使用豆瓣数据提供非豆瓣的购买链接，豆瓣可以在任意时间以任何理由终止API服务，豆瓣可随时要求删除特定或者全部豆瓣数据”等。

既当运动员又当裁判员。指平台提供者在平台里投入自己研发的产品，与第三方厂商竞争。这必然会加剧“二八原则”，使得资源越来越集中到少数者手里。比如据报道盛大游戏开放平台，几乎一半为自有产品。

产业链洗牌。由于资源组织的特性，开放平台一方面，不可避免使得竞争上升到“产业链竞争”，另一方面，它造成了互联网传统营销渠道的重大变革，从产品到用户的时间更加缩短了、方向更加精准了。正如当年新涌现的一批互联网“轻公司”通过建立全新的直营渠道，颠覆了传统的商品供应链，如今开放平台同样带来了新的产业重组。比如在百度的应用开放平台模式中，第三方的网络服务以应用模块的形式，可直接在搜索结果页面展现出来，实现“即搜即用”。鉴于百度巨大的互联网用户流量，无疑，这种模式将对下载、在线旅游、视频、杀毒、招聘等细分行业造成颠覆性冲击。

独辟蹊径是正道。对于有志创业的开发者来说，开放平台的热潮无疑是个历史机遇。但目前还远非“春天里”，据统计，即使叱咤风云的苹果应用商店，两年来也仅给开发者带来3000美元的人均年收入。很多开发者的现实体会是，其含辛茹苦开发出来的产品，上传到平台后，总在瞬间被汪洋大海般的同类产品淹没。回顾国内互联网创业的发展历程，不难发现：门户、电子商务、Web2.0、LBS……几乎每一波热潮，无不山寨西方模式，早期弄潮者依靠山寨的速度和效率获得巨大成功。然而随着“山寨”已经普遍经验，业内产生的任何一个新创意、一门新技术被山寨的概率越来越大、周期越来越短。因此，开发者们尤其在知识产权不完善的大环境下，只有立足长尾理论，不盲目跟风，专注自己的核心优势，做大公司不愿意做、小公司做不了的产品，才有脱颖而出的机会。

结束语

总得来说，目前中国软件的开放平台征途，尽管热闹，但还在初级阶段，还没造成对整个产业伤筋动骨的局面。真正激烈的平台鏖战，将发生在物联网、云计算技术进一步成熟，数字世界与物理世界的服务整合进一步密切与完善的时候。所以当前业界最需要的是，还是法律与行业规范的建设。

如果你试图更精微地考察开放平台的未来，记得目前中国的IT圈，正热传一部由《连线》前主编凯文·凯利撰写的科技预言巨著《失控》，该书指出，人类的一切智能创新都能在自然生物界找到原型。这说明新事物并不是无中生有，都是现实存在的某种映射。开放平台也是如此，它的发展脉络，也许就在历史典籍勾勒的轨迹里；它的未来特征，也许就蕴含你所住小区周围那些正在悄悄变化的菜场、超市里。

构建更开放的微博平台

■ 文 / 杨卫华

微博，作为一种可以通过网页、WAP、手机、短信等多种方式“随时随地分享身边的新鲜事儿”的服务，通过社会各阶层之间的互相关注，形成了一个巨大的社会关系网络。新浪微博开放平台将这个关系网络开放并分享给所有第三方应用使用。应用通过和新浪微博的集成，可以利用微博社交平台，增强用户之间交流，大大加强用户活跃度和应用黏性。

微博平台架构简介

在2010年11月的首届微博开发者大会上，新浪全面从技术和产品上全方位介绍了微博平台。经过几个月的发展，微博数据量和访问规模比当时已经有非常大的变化，本文根据最新的情况介绍微博平台在技术方面的改进。

从微博平台的需求来看，根据经验，Web 2.0平台发展到一定规模后，API访问量会逐渐超过主站本身访问规模，因此需要设计一个能支撑海量访问规模的平台系统。从应用的需求来看要求接口访问速度快、基于接口开发应用容易。从用户的角度，要求平台有良好的安全性，由于微博的用户大部分不懂技术，因此希望在平台上使用的应用是安全和可信的。因此微博平台在技术层面需要解决的主要需求有：满足大访问量、处理速度快、运行稳定，接口易用并且平台是安全的。

为了满足这些需求，从设计策略上，平台化和服务化是首要的策略。随着微博用户增长和业务复杂度的增加，业务服务化后可以结构更清晰，降低耦合，并可以独立考虑每个服务模块的性能、可扩展、安全等问题。服务化后，平台分成微博、用户、关系、计数为主的各个功能服务。每个服务独立考虑自身的业务逻辑及高速访问，如图1所示。

数据是微博架构非常重要的一个层面，数据包括存储及高速访问方案。在存储领域，MySQL依然是这个领域的王者，MySQL在稳定性和可靠性方面确实非常优秀。在数据拆分、业务扩容、failover方面也非常成熟。国外不少技术团队都喜欢使用新技术，但是从很多Google或Facebook出去创业的技术团队来看，MySQL貌似是他们不愿意舍弃的标配之一，今年新兴的互联网服务Quora技术负责人也建议首选MySQL：“在没有达到几百万用户之前，建议不要考虑NoSQL。”

MySQL虽然稳定和可靠，但是并不能满足高并发高性能的Web 2.0应用数据访问需求。微博系统中大部分业务数据都是海量规模且处于实时不断变化中，传统的cache方法并不能高效解决多个服务之间，cache实时更新的一致性问题。另外，传统的cache方案并不具备持久化能力，在cache故障或者重启之后往往会带来MySQL无法承载的访问量，从而进一步导致业务灾难。因此我们需要寻找一些新的更高性能的开源数据访问方案，比如Redis、MongoDB等新兴NoSQL产品。

Redis优势在于丰富的数据结构及高性能。比如微博的关系列表、业务计数等用Redis都可

作者简介：

杨卫华，新浪产品事业部技术经理，专注于开发高并发的分布式应用。对互联网后端技术，分布式，网络编程，XMPP即时通讯等领域感兴趣。曾多次组织广州及珠三角技术沙龙活动。

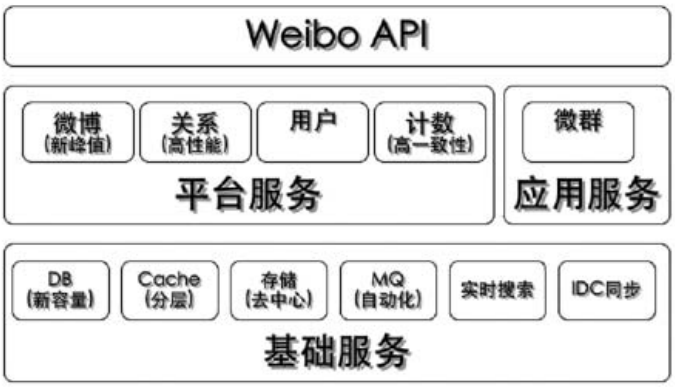


图1 微博系统架构图

以很方便地存储。Redis在常规的压力测试中性能已经超过Memcached，对于小规模数据量且需要高速访问的业务尤为适合。Redis目前成熟的存储模式是使用快照方式，它的限制是存储的数据规模需要小于服务器的物理内存。Redis写快照时是先fork一个子进程，利用了操作系统子进程复制（copy-on-write）父进程全部数据的优势来持久化数据。由于Redis是单线程的，fork时刻数据不会同时被其他网络访问修改，保证了数据的原子性及一致性。由于是子进程在进行持久化操作，写快照过程中父进程网络操作不受影响。AOF是Redis另外一种存储方式，它可以保证在服务器宕机时不会丢失数据，VM存储方式可以让Redis突破物理内存的限制，存储更多的数据，但是后两种在实际运行中存在不少问题，不推荐大型的应用使用。Redis最近新开发了一种diskstore的存储方式，它的目的也是希望突破物理内存的限制。diskstore原理和MySQL+Memcache方式类似，只不过将两者功能合二为一到一个底层服务中，简化了调用。读数据时使用read through以及LRU方式，内存中不存在的数据从磁盘拉取并放入内存，内存中放不下的数据采用LRU淘汰。写数据时采用另外spawn一个线程单独处理。由于VM方式实现复杂且稳定性不佳，diskstore是Redis作者以后重点发展的方向。

MongoDB是一种文档数据库，具备海量数据的支持能力，并且它还支持auth-sharding、复制、各种复杂高级查询等特性。MongoDB最大的特点是查询和插入性能非常快，部分情况下会比MySQL高一个数量级，另外还具有Schema-free的特性。比较而言，MySQL在海量数据情况下修改表结构是一项非常费时费力的工作，但在MongoDB中，应用程序可以随时改变存入的数据结构。使用MongoDB的另外一个好处和Redis有点相似，就是从存储到业务之间不需要再加cache层，从而简化了编程及开发效率。在评估MongoDB过程中，我们也参看了一些业界的案例，如Wordnik, 一个使用MongoDB的词典应用，这个系统每天的接口访问量为千万级。Mongo插入量可达到每秒8000以上，峰值可达每秒5万次，每个Mongo节点的数据量保存约3T数据。在微博平台中，目前尝试用MongoDB来存储一些需要经常变更结构、且需要高速访问的业务数据。

新浪微博API简介

Web服务有几种方式，传统的有Web Service和XML-RPC，但是从目前趋势来看，越来越多的开放平台（如Facebook, Twitter等）都使用REST模式的API接口。因此新浪微博平台也是使用REST模式。

目前新浪微博几乎所有功能都有对应API，而且接口数量还在迅速增长中。新浪微博API与其他平台差异性如下。

全面性。新浪微博提供各种全面数据访问接口，目前也有大量接口（如分组）在小范围开发者中邀请测试中，更多接口也即将开发完成并开放。

开发支持。新浪微博接口提供所有主流开发语言SDK，而且也处于不断改进中。另外在全国范围也陆续组织多次本地新浪微博开发者面对面交流活动，介绍微博平台最新特性及进行现场开发者技术答疑。

易用性。在第一版API设计中大部分是参考业界同行的设计思路，不过由于国内应用需求及网络环境差异，直接照搬国外经验未必完全合适，因此在后续版本API会根据开发者需求作很多改进，提供优先考虑开发易用的API。

新一代推送接口简介

REST API对于大部分开放平台都是够用

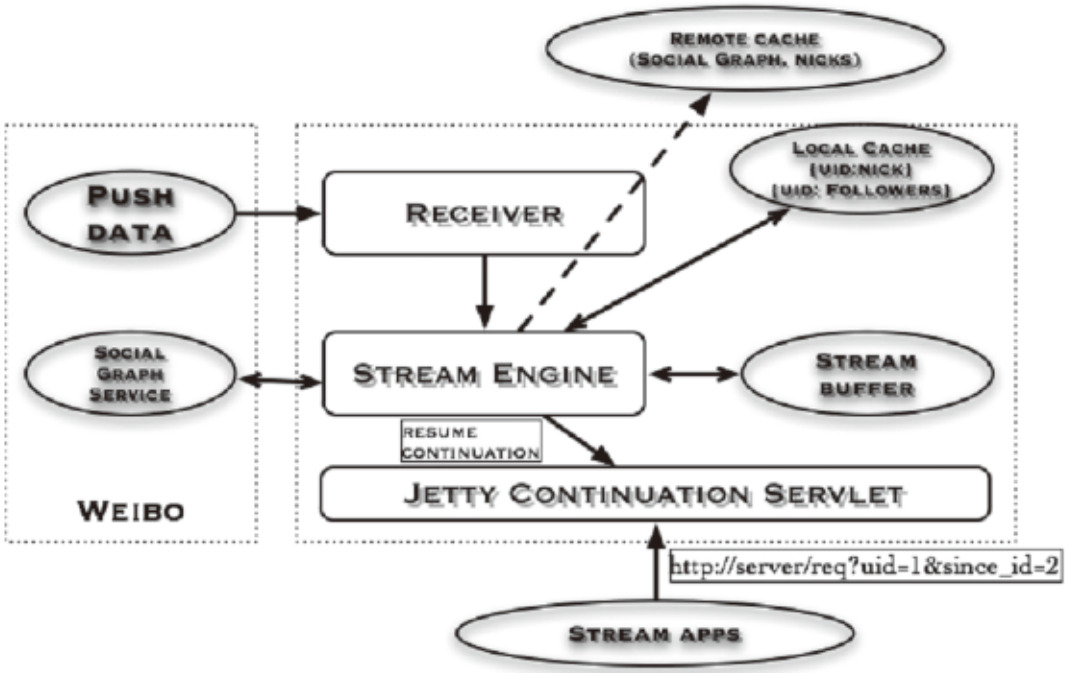


图2　推送接口的实现图

的，但是在微博平台上有一些不同的特点。微博接口的大部分请求都是为了获取最新数据，即获取当前用户的最新未读微博、评论、私信、通知等。从业务特点来看有点类似即时通讯系统，即时通讯软件通常是客户端和服务器之间通过长连及推送来实现即时数据投递，因此用REST API方式来获取微博最新数据就有很大缺点：

- 大部分调用都是空返回；
- 大部分时间在处理不必要的询问；
- 无法实时投递；
- 存在请求数限制（rate limit）。

因此微博接口也最好能提供一种推送协议，它采用推送的方式，有新数据服务器立即推送给调用方，无数据则不消耗流量，这样客户端实现也更简单。

在推送接口中，长连是一大技术难关。在图2中，使用了Continuation的方法，客户端连上服务器之后如果没有新的数据则处于挂起状态，几乎不占用系统资源，当有新的业务数据到达时，经过Stream Engine的业务逻辑代码定位到投递用户，Engine通过唤醒对应的HTTP连接并将数据实时投递到对应的客户端。推送接口中另外一大技术难题是需要按用户关注关系投递，由于关系是一个非常庞大的海量数据，而且处于实时变化的过程中，因此在图中使用了多层访问加速，首先会从最左边的Social Graph Service加载，加载之后会同时保存到推送系统的Local和Remote Cache中，以便实现推送系统高速访问。当然Social Graph数据的一致性也非常重要，需要增加一些机制来保证Cache一致性。

推送接口目前处于内部测试阶段，将会尽快对开发者开放。

开放，一直是新浪微博平台的宗旨。新浪一直希望能够通过开放平台实现开发者、用户和新浪自身的三方共赢。微博平台目前提供PHP、Java、Android、C#、ActionScript、iOS、C++等多种SDK支持。希望更多的开发者加入到新浪微博开发者行列。

人人网开放平台浅谈

■ 文 / 李志才 李福松

作者简介：
李志才，人人网开放平台API技术主管

作者简介：
李福松，人人网开放平台高级经理

人人网于2008年7月8日正式对外发布了开放平台战略，从此全面拉开了中国互联网的开放平台时代。开放平台成为创新者和创业者的乐土，各种开发团体、个人、开发公司都进驻开放平台，各种应用产品五花八门，极大地满足了用户的需求，涌现出了很多优秀的应用产品。像风靡世界的开心农场就是从人人网开放平台率先推出并迅速受到广大用户喜爱，一时间全民挖菜，场面蔚为壮观。国际上颇具影响力的著名IT科技杂志《连线》评选出了“过去十年最具影响力的15款游戏”，开心农场排名十四。在收获优秀应用产品的同时，开放平台也帮助众多开发者迅速创业成功。

下面将从平台架构、授权机制、API、人人小部件等几个方面介绍一下开放平台。

平台架构

人人网开放平台是一个通用的开放平台，具有灵活的架构，能够支持站内应用、站外Web和WAP网站、桌面应用，以及iPhone、Android等移动终端上应用的接入。而且，人人网开放平台的架构正向着标准化和简洁易用的方向发展。

在人人网开放平台的技术架构中，验证授权和人人API是最核心服务。验证授权，是为了识别用户身份并引导用户对第三方应用所要求的访问权限进行处理，用户可以同意或拒绝授予第三方权限，这样就在很大程度上保证了用户的隐私安全。人人网开放平台的验证授权服务，实现了业界最先进的OAuth 2.0协议，它与现存的其他授权方式相比，在适用范围和易用性上都有了非常大的提高。下面会有专门的章节进行介绍。

第三方应用在获得用户的授权后，就可以调用人人API了。通过人人API，第三方应用能够完成各种各样的功能，如获取到当前用户的个人信息、好友关系等，还可以完成很多以前必须在人人网上才能完成的动作，如发表状态、发表日志、上传照片等。人人API采用RESTful的方式以HTTP协议提供服务，使得它的调用方式简单统一，能够适用于任何网络环境。

与人人网的紧密集成

在人人网上开发的应用，可以与人人网的用户功能紧密集成，例如，第三方应用可以在人人网提供的一个容器（apps.renren.com）中运行，并可进入左侧菜单，让用户感觉第三方的应用就是一个真正的站内应用。这样的紧密集成，在很多微博类的开放平台中是不具备的，它能够极大地提升用户体验。

为了支持这样的紧密集成，站内应用分为两大基本类型：XNML类应用和iframe类应用。

XNML的全称是XiaoNei Markup Language（校内标记语言）。在XNML类应用中，除了上述的验证授权和人人API这两项基本服务之外，还提供了XNML标签解析服务。当用户浏览器请求apps.renren.com容器时，人人网代理服务器向第三方服务器请求一段XNML代码，这样的代码由HTML标签、CSS、JavaScript以及经过扩展的XNML标签组成。一个XNML标签类似于<xn:name>这样的形式，使得第三方开发者在应用中集成人人网的功能异常简单。当XNML代码返回时，中途被人人网XNML标签解析服务进行渲染，成为真正的HTML代码返回给浏览器显示。

由于第三方返回的XNML代码中存在JavaScript和CSS，因此人人网XNML标签解析服务器对这些JavaScript和CSS进行了安全过滤，过滤后的代码能够安全地运行在人人网的站内应用容器中。这种在XNML类应用中使用JavaScript和CSS的技术分别称为XNJS和XNCSS，系统架构如图1所示。

第二大类站内应用称为iframe类应用。这类应用是直接在人人网的站内应用容器页面中嵌入一个iframe，用iframe的src指向第三方服务器地址。这种集成方式调试更加简单，与用户信息的集成主要依靠前面讲到的人人API技术；但

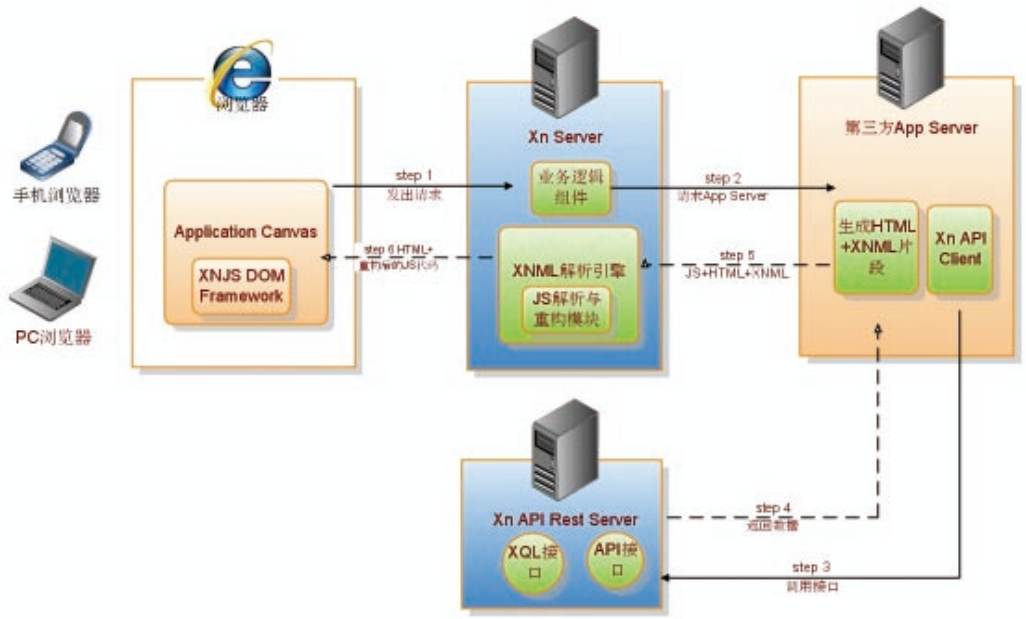


图1　XNML类系统架构

缺点就是无法再使用简单易用的XNML标签。为了解决这一矛盾，人人网开放平台为iframe类应用开发了EXNML（扩展的XNML），允许第三方开发者能够在iframe类应用中仍然使用类似于XNML的标签。另外，EXNML也能够被第三方人人连接网站使用。

人人连接架构

人人连接技术允许用户用人人网账号登录第三方网站，进行帐号绑定，分享精彩内容，并携带社会化图谱进入第三方网站，与好友在第三方网站上深入互动。

人人连接基于OAuth 2.0授权技术，在用户验证和授权完成之后，以一种安全的方式将用户会话信息传递给第三方网站。由于用户在renren.com域名下进行验证和授权，与第三方网站处于不同的域名，因此需要跨域的技术来传递信息。跨域技术，在人人连接的技术体系中占据非常重要的地位。实现跨域的技术方案多种多样，如JSONP、iframe、Flash、postMessage等机制。

人人连接还实现了一整套的JavaScript SDK，允许第三方通过JavaScript调用就能够完成获取用户和好友信息，发送自定义新鲜事等功能。另外，JavaScript SDK还通过页面渲染的方式能够识别EXNML标签，以类似于XNML的方式来调用功能。

人人连接是一个宽泛的技术概念，它还包含了人人网开放平台提供的诸多小部件（Widget），这里面有人人喜欢、实时讨论、粉丝部件、好友派等。这些Widget是一种嵌入式的小插件，第三方可以以iframe的方式随意将它们嵌入到网站中去，具有简单易集成且功能丰富的特点。其中人人喜欢是一项重要技术，它遵循Open Graph协议，允许用户把任何他所喜欢的站外网页和资源当作一个人人网的公共主页来对待，便于日后追踪和获取动态更新。

未来的平台架构

人人网开放平台的技术架构将向着标准化的方向发展，层次划分更加清晰，对第三方的开放程度不断加大。

人人网开放平台提出了通用编程平台（Common Programming Platform，简称CPP）的概念，其体系架构如图2所示。

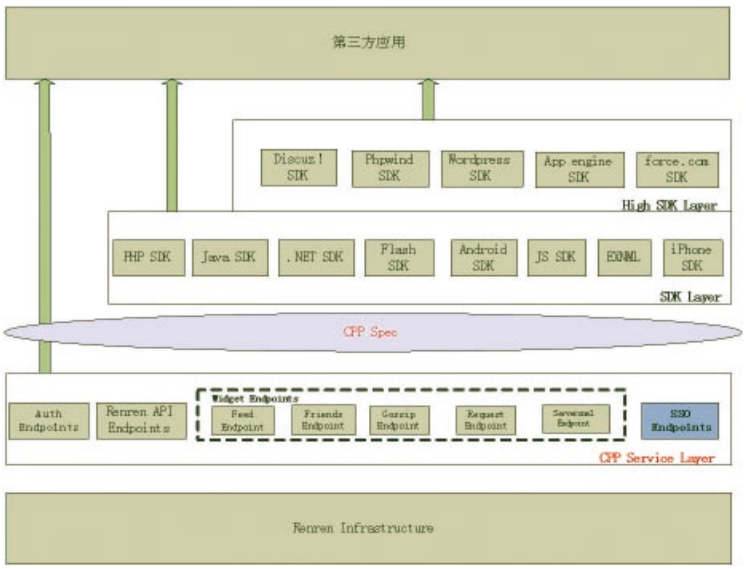


图2　CPP体系架构

在CPP的平台架构下，平台的核心由底层的一个CPP Service层来实现，在这上面有严格定义的规范（CPP Specification）。基于这个CPP规范，开发人员（包括平台和第三方）可以在上面封装各种SDK。CPP未来平台架构将具有如下特性。

- **通用性**。更好地支持Web、Mobile、Desktop等环境。
- **标准化**。在服务（Service）层次上标准化，利于第三方获得稳定清晰的编程环境。
- **行业分工的清晰化**。标准化带来的好处之一，是可以让第三方在开放平台行业发展中发挥更大的作用，即SDK的开发维护工作可以交给第三方开源社区来完成。开源社区在通用编程平台的Service基础上开发SDK，第三方应用开发商在SDK基础上开发应用。
- **技术架构的清晰化**。利于平台对服务进行维护，也利于第三方对平台架构的理解。
- **扩大开放范围**。新的架构在某些出于安全考虑而无法通过RESTful开放的功能方面，可以达到与RESTful非常接近的通用效果。

授权

授权是指用户授予第三方应用某种权限。用户在使用第三方应用时，第三方经常需要知道当前用户的ID、名字、头像、好友关系等，用户的这些数据保存在人人网上。只有当用户同意后，人人网才能将这些数据提供第三方。这个同意的过程就是授权。以下是一个典型的授权页面。

经常会遇到人们问我：为什么需要授权？其实，用户使用人人网，把自己的姓名、照片、日志等发表在人人网上，这些资源的所有者是用户，不是人人网，人人网只是负责存储和展现。我们充分尊重用户，没有经过用户的同意，不能把这些资源任意给第三方。用户同意的过程就是授权，只有等用户授权后我们才能通过人人API把用户的资源给第三方。

OAuth2.0

人人网于2011年1月底正式推出了OAuth2.0。OAuth是个统一的授权协议，它更安全、更通用。适用于Web站点、桌面软件、手机WAP站点、手机应用程序，甚至是一些受限的设备，比如能上网的电视、电冰箱等。国内一些网站目前支持的是OAuth1.0，作为国内首家支持OAuth2.0的网站，我们同时完成了OAuth2.0中文文档的翻译工作。通过OAuth2.0，无论是什么类型的应用，也无论是什么设备的应用，均可以接入人人网，使用我们的API。

目前人人OAuth2.0目前支持两种验证授权流程，Web Server Flow 和 User-Agent Flow。

Web Server Flow适用于有Server的应用，例如Web站点、有Server的客户端应用等，它的工作流程如下。

1. 获取Authorization Code。
2. 使用Authorization Code换取Access Token 。

User-Agent Flow适用于无Server的应用，例如桌面程序、手机客户端程序、浏览器插件等，可以直接获取Access Token。

一旦应用获得了Access Token，就可以调用人人API来获取数据。

人人API

当应用获得了用户的授权之后，应用就可以通过人人API来进行数据的存取，人人API以HTTP远程接口的形式提供。所以，无论你用什么开发语言，只需按照HTTP协议向人人API服务器发送请求即可调用，同时为了简化开发者的调用，还有很多种不同语言版本的SDK包可供选择，如 Java SDK、PHP SDK、.NET SDK、Python SDK、Ruby SDK、Perl SDK、JavaScript SDK 等。

人人API功能丰富，通过API可以获取用户的姓名、头像、性别、家乡等基本信息，也可以取得他们的好友列表、应用好友等，还可以阅读日志、发表日志、浏览照片、上传照片、发表状态、留言等，甚至可以发好友请求、分享精彩内容、赠送礼物等。

人人API会开放更多的API给第三方使用。借助OAuth2.0，在互联网的各个领域，都可以使用人人API来开发应用。在技术上会进行创新，提供新的API调用方式，支持更多类型的返回值。我们要通过一些技术变革，一来提升服务性能，让API服务更快更稳定；二来简化调用方式，使开发者更易理解、更易上手，降低学习和使用成本；三来要重新整理下文档，提供适合各种水平开发者的文档。

人人小部件

为了让人人网用户能够随时随地享受到人人网提供的各种服务，能够方便地和好友继续保持着沟通，开放平台着力打造了我们称作“人人小部件”的一套产品。这就好像散布在世界各地的中餐馆一样，无论你身在世界的哪个国家，你都能够找到中餐馆品尝到家乡的美味。

人人小部件是一套产品库，包含了如下的产品。

分类	名称	集成方式	特点
分享工具	人人喜欢	iFrame方式嵌入到目标页面	新一代的分享工具，产品链更完整
	UGC分享	嵌入Html代码到目标页面	非常简单，形式多样，被广泛采用
	自由分享	嵌入Html代码到目标页面	同上，且支持计数器显示
	人人连接分享	嵌入人人XNML代码到目标页面	需要事先注册成为平台应用
登录按钮	好友登录	嵌入人人XNML代码到目标页面	需要事先注册成为平台应用
社会化部件	好友派	嵌入人人XNML代码到目标页面	需要事先注册成为平台应用
	实时讨论	iFrame或XNML方式嵌入到目标页面	满足用户实时交流讨论
	粉丝部件	iFrame方式嵌入到目标页面	将公共主页的粉丝显示在站外

人人喜欢

用户在第三方网站看到希望分享的内容都可以点击“喜欢按钮”，将内容分享给他的人人网好友，好友将会在新鲜事中收到这种分享信息，同时也有机会进行二次分享传播。如果内容有对应人人网公共主页，则用户同时可以成为该公共主页的粉丝，以后就可以持续地接收到来自该主页的最新动态。

对于用户来讲这种操作非常简单；对于网站来讲嵌入这种“喜欢按钮”的开发成本也非常小，只需要嵌入一段iframe代码，同时相应的技术支持文档非常丰富。一旦网站拥有了人人网公共主页，则可以持续地利用公共主页进行运营工作，和粉丝保持联系，准确进行消息投放。

由于人人喜欢是基于Open Graph协议，人人网在展现人人喜欢产生的新鲜事时会尝试从第三方网站的页面中抓取合适的信息来组织新鲜事的格式与内容，所以第三方网站如果也依据Open Graph协议在网页<head>中嵌入一些<meta>标签，那么人人网就可以准确抓取目标网页的信息，从而新鲜事样式更准确，内容归类更明确，方便后期聚合和分类标准。下面的表格是Open Graph中标签属性定义：

标签属性	描述
og:title	标题
og:type	类型，类型请使用协议官方网站上Object types定义的类型
og:image	表示页面信息的一张缩略图地址
og:url	代表页面的规范url
og:description	关于页面信息的简短描述
og:site_name	页面所在的网站名称
og:videosrc	视频Flash地址
og:audiosrc	填写音频的.mp3地址
xn:page_id	填写需要关联的公共主页ID

图3　人人喜欢的格式与Open Graph的对应关系

图3是人人喜欢产生的新鲜事格式与Open Graph部分属性的对应关系。

实时讨论（Live Stream Box）

这是一个复合性的产品，支持发表状态、展示好友状态、回复状态，所以可以被广泛地嵌入到很多网页中，用以满足用户发表与交流的需求。较为典型的应用场景是针对某一话题展开实时讨论，例如在某些网上直播赛事、线上活动中供网友交流，也有一些Social Game嵌入它作为一种长期的玩家交流工具。

它支持采用iframe的方式嵌入到目标网页中，开发成本较低，同时也支持作为EXNML标签嵌入到目标网页。

这两年，国内互联网异常活跃，各种类型的开放平台也如雨后春笋般推出。人人网开放平台坚持既定的开放策略，规划着自己的未来发展之路，更大程度地开放，更大程度的满足多种需求，更大程度地与开发者们共赢。我们清楚地认识到，要做到这些目标，就应该将开放平台的各种基础服务夯实，将各种流程优化，通过不断的技术创新来保证开放平台的可持续发展。

以上就是对人人小部件的简要介绍，更详细的信息可以访问人人网开放平台(http://dev.renren.com)。

腾讯微博开放平台解析

■ 文 / 袁清 申婷

作者简介：
袁清，研发工程师

作者简介：
申婷，研发工程师

什么是腾讯微博开放平台？

腾讯微博开放平台，为广大开发者提供开放接口，可构建丰富多样的应用。开放平台为用户提供辅助开发的多种典型应用，让腾讯微博通过第三方网站、用户博客、签名档等形式遍植整个互联网，并提供Qweibo系统协助第三方快速搭建起自己的微博系统，并与腾讯微博平台互通，充分利用腾讯用户与信息资源。

第三方应用能从微博获取海量资讯，或将信息传播到千万级用户的平台中，得到营销推广机会；利用腾讯微博开放平台提供的数据分享和传播服务，加上开发者的智慧，将创造无穷的功能与乐趣。

腾讯微博开放API介绍

腾讯微博开放API已向开发者提供以下功能接口，开发者可利用这些API开发自己的微博应用：

- 用户授权
- 微博读写
- 收听关系
- 搜索（用户/微博）
- 话题/私信/收藏

使用腾讯微博为第三方开发者提供的以上API，可以获得如下好处：

- **强大的平台支持**，腾讯微博拥有8000万注册用户，这些用户都可能成为第三方应用的用户群体。
- **获得来自腾讯微博的优质内容**，微博天然具有内容即时丰富的特点，可以通过名单\话题\搜索等，将腾讯微博的信息输出到第三方应用，获得最即时、最丰富、最活跃、最热门的微博消息，紧急突发事件，提供来自现场的最新鲜资讯。
- **内容通过腾讯微博进行审核**，API全部内容通过腾讯微博审核，第三方应用无审核压力，专业的审核团队24小时待命。
- **降低网友参与门槛，提高网友参与度**，微博内容简短，发布门槛低，天然有利于用户的加入，相比传统论坛、博客等形式，微博更有利于提高网友的活跃度，微博具有的转播、评论、对话、私信等功能，有利于网友之间产生互动。
- **获得在腾讯微博进行品牌营销的机会**：

(1) 优秀的第三方开发者的官方微博，在腾讯获得认证及推荐，在显著位置得到曝光，增加听众。

(2) 第三方应用产生微博来源显示第三方网址及网站名称，不仅能为网站带来点击，同时也能增加在亿万级平台之上的曝光机会。

典型应用类型

使用腾讯微博开放API可实现丰富的微博应用形式，包括：

1. **账号连接**——使用腾讯微博帐号登录/绑定腾讯微博帐号，分享信息；
2. **个人展现**——通过微博签名/收听按钮/广播站等形式，展现用户个人/微博信息，进而产生用户间的关系链；
3. **信息聚类**——话题/名单/用户发表的微博聚类展现；
4. **用户互动**——网站内容（网页/图片/视频）发表到微博，获取本站产生微博信息点评，回馈到网站作为用户产生内容；
5. **小工具、小游戏**——更多精彩的应用形式，还等待各位开发者加入开放平台进行创造。

开放平台已经提供的服务

1. 使用腾讯微博账号登录第三方网站，通过覆盖率超高的腾讯账号，有效降低第三方网

站用户参与门槛。

2. 授权使用腾讯微博帐号，将在第三方网站中的动态，通过微博消息的形式，自动发表到腾讯微博平台中，让更多微博用户通过消息中的链接/来源，回到第三方网站。

3. 第三方在腾讯微博中运营 **话题/名单/用户**，可通过开放API，查询腾讯微博平台中针对某个话题/名单/用户的微博，进行**集中展现**，达到获取微博平台信息**作为网站内容**，并**吸引用户参与**的目的。

4. 微博签名档：可用于邮件/论坛等处，展现最新一条微博。

5. 微博广播站：可展现个人信息+最新微博信息+发微博窗口，用于个人网站/博客等处。

6. 收听按钮：点击收听此用户，可植入网页任意位置。

7. 数据服务：开放平台可为第三方应用，提供消息读写数据，用户授权/登录数据，使用一键转播功能，在腾讯微博每日产生消息数等数据，协助第三方了解自己的运营情况，并根据数据改进自己的产品。

技术架构介绍

腾讯微博采用标准OAuth协议接入。

OAuth是一种开放的协议，为桌面程序或者基于BS的Web应用提供了一种简单的，标准的方式去访问需要用户授权的API服务。OAuth类似于Flickr Auth、Google的AuthSub、Yahoo的BBAuth、 Facebook Auth等。OAuth认证授权具有以下特点：

1. 简单：不管是OAuth服务提供者还是应用开发者，都很容易于理解与使用；

2. 安全：没有涉及用户密钥等信息，更安全更灵活；

3. 开放：任何服务提供商都可以实现OAuth，任何软件开发商都可以使用OAuth。

第三方应用每次向OAuth三个服务地址发送请求时，必须对请求进行签名。签名的方法有：HMAC-SHA1、RSA-SHA1与PLAINTEXT等三种。腾讯微博开放平台暂只支持HMAC-SHA1。

目前，OAuth2.0草案正在制定中。相比1.0而言，最主要的变化是增加了 access token “refresh”的概念，即access token不再是永久有效的，如果较长时间不使用则会过期。此时需要用户“续期”才能继续使用。

腾讯微博开放平台遵循最新的RFC 5849协议。

架构上主要分为应用层、逻辑层、数据层三大块，可满足腾讯数亿用户的访问需求。

应用层主要是开发者身份申请，APP_key申请，以及一件转播、微博秀、广播站、签名档等官方应用；

逻辑层主要包括放号模块server，授权server、微博对外API接口、鉴权server四部分；

存储层主要包括分布式数据存储模块、微博数据接口模块。

具体结构如图1。

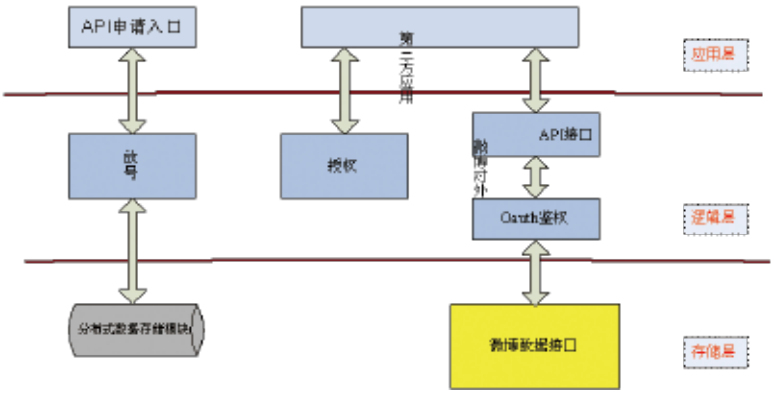


图1　腾讯微博平台架构

具体的用户授权流程如图2：

第1步，用户点击连接纽，触发授权过程；

第2步，第三方软件向OAUTH服务提供商请求未授权的request_token。向request_token URL发起请求，请求需要带上的参数：（app_key、signature_method、signature、times_tamp、nonce、version、callback_url）其中signature由第三方对base_string（由各个参数组成的字符串）通过使用app_secret密钥，经过signature_method（HMAC-SHA1,RSA-SHA1）方法进行签名；

第3、4、5步，验证第三方请求合法性，服务方用app_secrect密钥base_string进行签名，并验证得到的签名与第三方传过来的签名是否一致；

第6、7步，OAUTH服务提供商同意使用者的请求，并向其颁发未经用户授权的request_token与对应的request_token_secret，并返回给使用者；

第8步，第三方重定向到服务方页面，让用户进行授权，此步请求通过GET请求，带上request_token，此请求未签名；

第9步，OAUTH服务提供商将引导用户授权。该过程可能会提示用户，你想将哪些受保

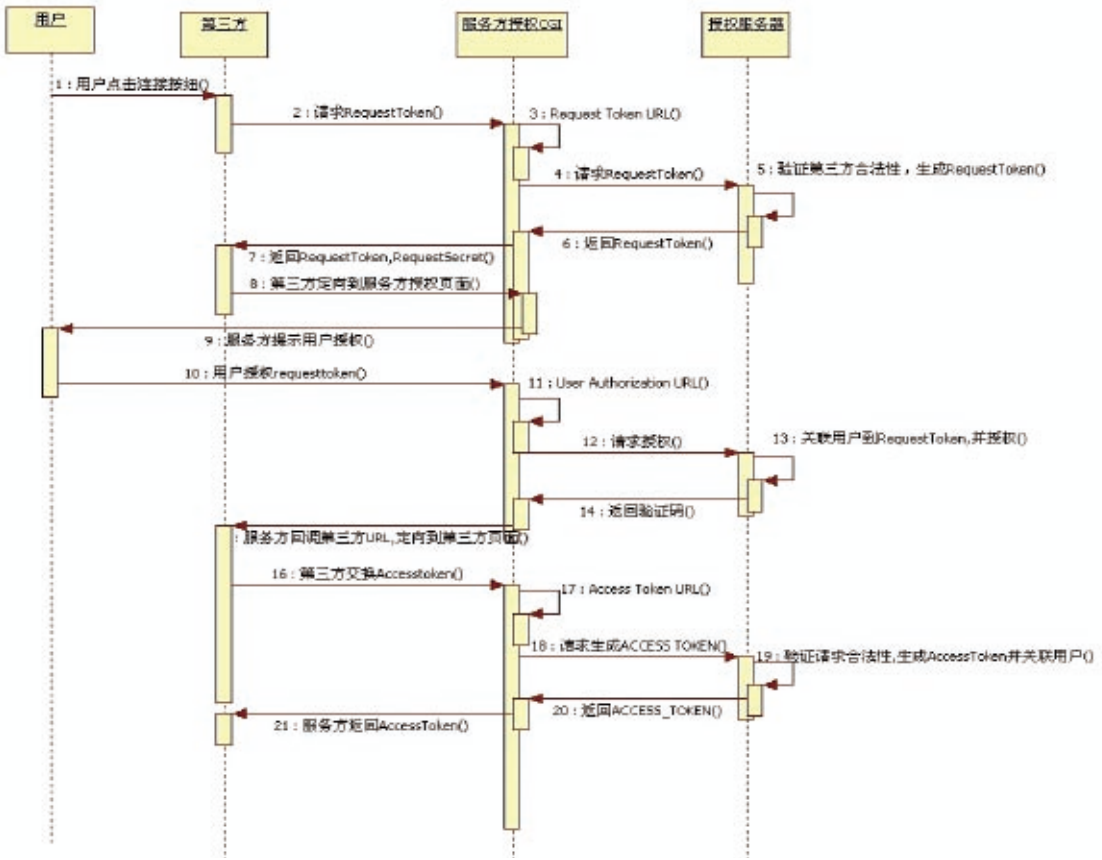


图2　用户授权流程图

护的资源授权给第三方应用；

第10步，用户点击授权按钮，提交授权请求到User Authorization URL；

第11、12、13步，服务方验证request_token有效性（是否过期，是否已使用过），如果有效，关联request_token与用户ID；

第14、15步，服务方回调第三方的callback_url，带上request_token，及verify_code。verify_code的作用，用于确认下步请求与回调对方是否同一用户；

第16步，第三方得知request_token 授权后，第三方将向access_token URL发起请求，将上步授权的tequest_token换取成access_token。请求的参数，比第2步多了一个参数就是request_token，（app_key、request_token、signature_method、signature、times_tamp、nonce、version）其中signature由第三对base_string(由各个参数组成的字符串)通过app_secret加上request_token_secret作为密钥，经过signature_method（HMAC-SHA1,RSA-SHA1）进行签名；

第17、18、19步，验证第三方换取access_token请求合法性，服务方用app_secrect与request_token_secret密钥对base_string进行签名，并验证得到的签名与第三方传过来的签名是否一致；

第20、21步，OAUTH服务提供商同意第三方的请求，并向其颁发access_token与对应的密钥（access_token_secret）。

开放平台的发展方向

腾讯微博2010年4月正式上线。2010年7月考虑开放第三方平台，9月内部系统联调通过。11月第一个合作伙伴应用正式推出，到现在发展势头良好。其发展历程如下：

第一步：对第三方开发者提供API服务，可开发第三方应用；

第二步：搭建应用频道，面向第三方应用用户提供应用推荐及管理系统，面向开发者提供应用基本数据查看，开始对第三方应用提供运营帮助（目前处于此阶段）；

第三步：为开发者提供全面的技术与运营服务，包括作为推广平台的影音频道第三方APP的运营管理，及面向对第三方开发者，提供全面的应用数据查看及分析系统，并考虑为第三方应用提供服务器托管，通过提供技术咨询服务，以及组织开发者大会等形式，帮助第三方应用更好的发展，最终形成双赢的局面。

漫谈51开放平台的后台服务支持

■ 文 / 赵宏华

随着互联网的发展，目前国内已有数十家社区类网站准备或已经开放出自己的API，并且也正在渗透到搜索、电子商务等领域。

51.com在2008年6月底正式推出了开放平台，接受第三方开发者申请，并在同年的10月，将51币支付系统正式接入到开放平台中。这意味着第三方开发者可以结合该系统，开发各类能够通过虚拟货币支付的小应用。谈到51开放平台的服务支持，就不能不先提到其基础架构。简单地说，51开放平台和其他网站一样，采用的也是多层架构，本文将简单介绍51开放平台后台服务的发展现状及技术架构实现方法。

接口服务

首先我们看处于最上层的接口服务层，这一层用于第三方开发者和51开放平台做最直接的交互。我们的API采用了REST规范。REST首次出现于2000年Roy Fielding的博士论文中，近些年在一些流行的开放平台框架中频繁应用，它包括了以下的设计概念和准则：

- 网络上的所有事物都被抽象为资源（resource）
- 每个资源对应一个唯一的资源标识（resource identifier）
- 通过通用的连接器接口（generic connector interface）对资源进行操作
- 对资源的各种操作不会改变资源标识
- 所有的操作都是无状态的

当应用需要从开放平台获取或写入数据时，所有的API都是通过HTTP的POST方式向51API REST Server发送请求来实现，服务端返回XML或JSON等格式化的结果。因为所有的操作都是无状态的，所以REST Server可以做到横向扩展，不需要改变原来的架构。同时因为所有的数

作者简介：

赵宏华，51.com事业部副总经理，对系统架构、大型服务器后台等方向有深入经验，热爱编程，喜欢亲自实践。近期主要从事技术架构、基础后台、桌面及无线客户端等方向的开发管理工作。

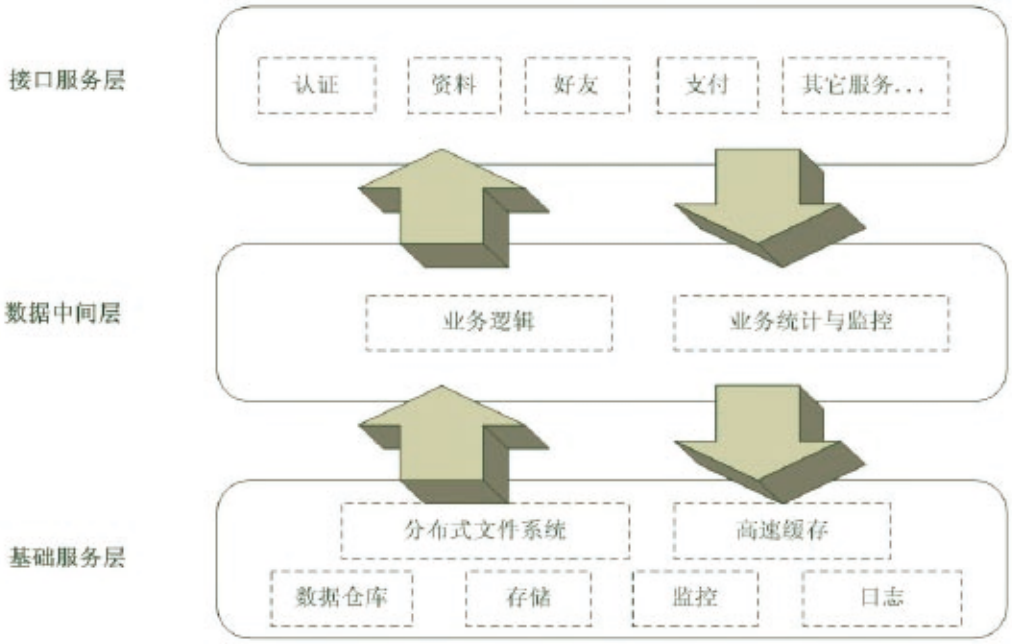


图1 51开发平台基础架构图

据都是格式化的，所以开发中的调试与测试都会变得非常简单。

高性能cache服务

一个成功的开放平台，上面会运行数以千计甚至万计的第三方应用，有些应用实时性比较强，对性能的要求自然会非常高。由于REST接口的无状态和第三方应用的独立性，使得在客户端上实现高性能、高可用的cache服务非常困难。所以开放平台的服务端一定要采用高性能cache系统，来满足第三方应用的需求。

我们目前采用的高性能cache主要有两种，一种就是目前被广泛应用于大负载高并发网站的Memcache，特别对于像51.com一样的社交网站，如Facebook、开心网、人人网等。由于这类网站用户数据相对庞大复杂，前端应用种类繁多，对服务器的响应速度要求相对比较高，频繁的数据库读写给系统带来的压力是难以想象的。Memcache的几个显著优势，注定了它是开放平台上最优先考虑使用的cache系统。

- **高性能**：Memcache内部实现是基于key-Value的内存Hash表，由于所有的操作都在内存中进行，具有极高的效率。
- **分布式**：开发人员可以通过部署任意数量的Memcache服务器，通过它们的组合可以达到非常大的cache容量，理论上只和物理服务器的数量和内存容量有关，非常易于扩展。
- **易使用**：接口丰富，第三方应用开发人员或其他Web开发人员可以非常方便地使用自己熟悉的语言来使用。

Memcache使用的是类似STL中小块内存分配类似的机制，即 Slab Allocator，基本原理是按照预定的大小，将内存分割为特定长度的块，它的好处是可以解决内存碎片问题，如图2所示。

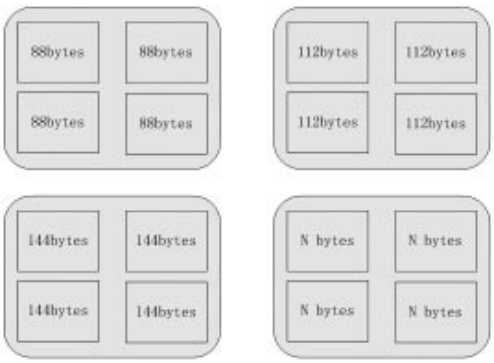


图2 Memcache使用的Slab内存分配示意图

但在我们的使用中，也发现了Memcache的一些缺陷，比如内存按块大小递增的固定大小分块，存在一定的内存浪费，对于一些cache对象的大小频繁或突然大量变化时就更为明显，甚至会出现内存有大量剩余但分配不出可用内存的情况。Memcache也存在系统设计的瓶颈，比如只能存储某个大小以内的对象。对于这些情况，我们的基础架构部重新设计并研发了自己的内存cache系统，新的系统吸取了Memcache部分优秀的设计经验，并针对其缺陷做了不少改进，实现了如下特性。

- 支持多变的数据规格，当数据对象的大小发生变化时，系统仍能正常工作。
- 内存cache模块和网络模块分离，可作为独立模块嵌入到C++应用服务器中。
- 内存不足时，系统可对内存空洞进行整理，各个不同块大小的内存分配器之间可以实现内存复用。具体效果如图3所示：

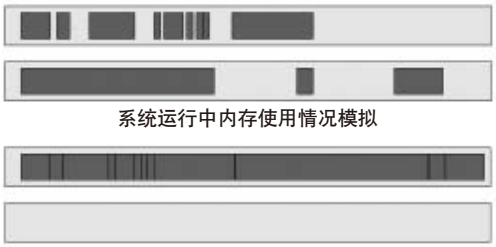


图3 51cache内存整理示意图

新系统对资源占用、使用限制上均有较大的改善。

海量数据存取服务

开放平台上存在着海量的数据存取，第三方应用需要从开放平台上获取和写入数据。对SNS网站而言，这种情况尤为突出。用户的个人资料、留言、相册、好友等都可能会被第三方应用来进行存取，如果没有一个强大的海量数据存取系统，平台数据的运营、维护将变得极为困难，最终会影响用户体验。为此，我们把数据存储系统分为了两个模块，第一块是基础类数据，比如个人资料、留言、好友列表等数据。第二块是文件类数据，比如照片、网盘等数据，并由此划分为两个子系统。

对于基础类数据，通常的做法都是存储于MySQL等关系型数据库中，前端通过拼装SQL语句来进行数据的读写，事实上51.com发展初期也都是这样做的。随着用户和业务的增长，数据库越来越难以负荷查询量的快速增加，于是我们开始对数据库进行拆表、拆库等操作。拆分的标

准首先是业务，然后就是垂直和水平的拆分，拆分时要停止数据库的写操作，对业务也会造成比较大的影响。每次这样的操作对项目组的开发人员都是一次伤筋动骨的过程，动辄耗费几个小时甚至一天的时间来做这些重复性的工作，到了后期，数据库的压力随着用户增长以几何级数的速度上升，拆库、拆表已经完全无法满足业务增长的需要，此路已经不通，只能考虑其他方案来解决这个问题。

在此情形下，我们设计并开发了中间层系统。前端的接口不再直接读写数据库，而是通过中间层系统来间接操作数据，中间层屏蔽了存储的细节，在底层存储发生改变时，可以做到对前端透明，同时在后端采用了前面所提到的高性能cache。对于一般的业务，读操作要远远大于写操作的情形下，这样能有效降低数据库的实际查询数。经过测算，一台普通的16GB内存的x86 64位服务器，在cache灌满时，前段的海量数据访问在cache上的命中率可以达到99.5%以上，这样就只有0.5%的数据读取，数据库的压力就可以大大降低。

对于文件类数据，我们考察了EMC、GFS、Hadoop等系统的设计思想和理念，研发了一套安全、高性能、高可用、可伸缩的分布式文件系统，以满足日益增长的文件存取需求。系统架构图如图4所示：

整个系统由MDS、OSS、MGS及数据库、文件存储等部件组成，其中MDS提供元数据的高速存取，负责系统集群内所有机器的负载均衡、调度管理及数据迁移等，OSS负责管理文件的数据存取，MGS是集群对外的管理系统。

一致性Hash

在分布式服务部署的时候，通常的做法是按照某个值（如用户账号、照片ID号等）为Key，做一个Hash算法，按照计算出的结果分布到服务器集群中的某一台确定的机器上。在我们的实际应用中，发现这种方式存在很大的弊端。在服务器集群节点发送变化时（如某台机器意外故障，一时难以恢复），一般的Hash方式会造成大量的数据节点移动，在cache系统上这种情况表现得更为明显，可能导致几乎全部的cache数据无法正确命中，后端存储系统压力突增，甚至难以支撑正常的业务。这种情况下，我们采用了一致性Hash分布来解决这个问题。

如图5所示，服务器节点均匀分布在一个

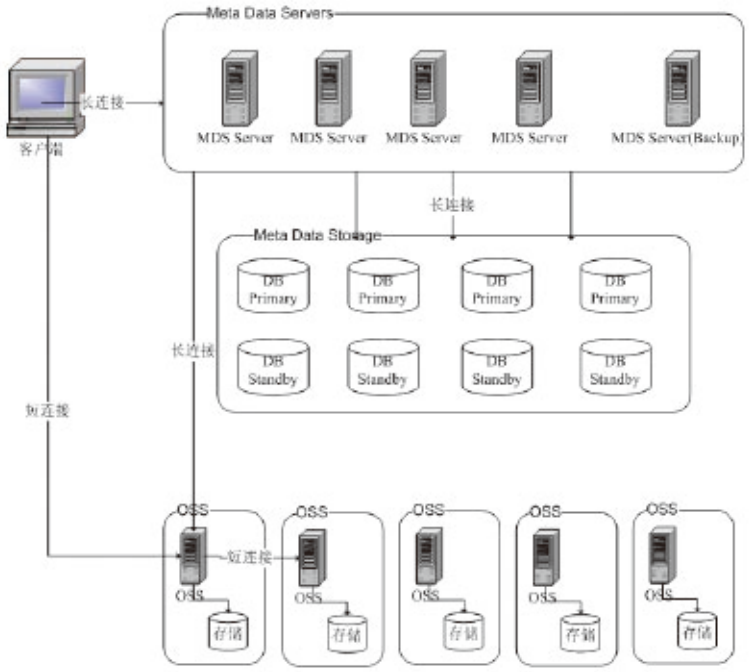


图4 51DFS系统架构图

0~232的圆环上，cache数据存入时，首先计算出在圆环的Hash值，然后存储到顺时针下一个最近的服务器节点上。当节点数目发生变动时，如node2发生故障，原先存储到node2节点上此时将分布到node4上。同理，当圆环中增加一个服务器节点时，原先存储到相邻节点的数据此时将分

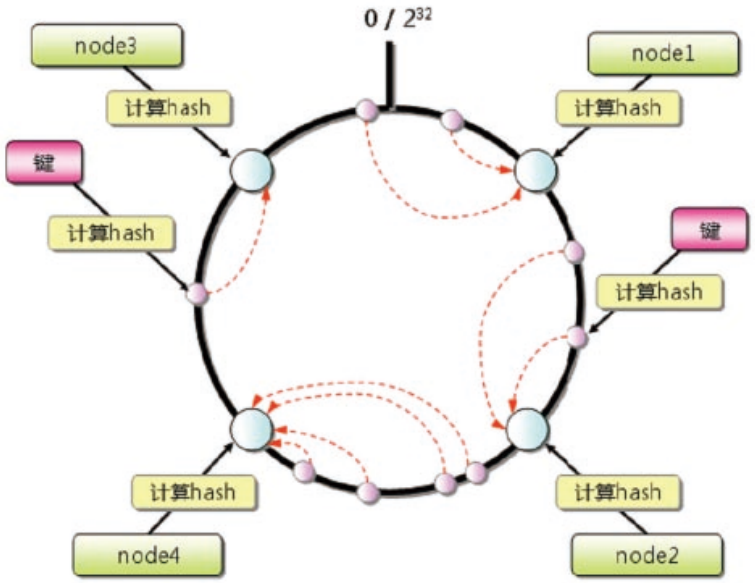


图5 一致性Hash原理

出一部分到新增的节点上。可以看出，采用一致性Hash的数据分布策略后，发生事故或者系统扩容时，能大大降低系统整体性的数据迁移和变动。

逐步改善，设计优秀API

■ 文 / Jaroslav Tulach

判断一个API是否优秀，并不是简单地根据第一个版本给出判断的，而是要看多年后，该API是否还能存在，是否仍旧保持得不错。

第一个版本远非完美

第一个版本总是来得特别容易，不仅容易开发，而且容易发布。API的需求会随着时间而变，那些过去有效的API可能现在已经不再适用了。而且每个程序中都会存在Bug，需要不断地来修复，这样做带来的副作用人所共知：修复一个Bug的同时会引入两个新Bug。这些观点普遍适用于所有软件系统，API也不例外。

但我们没必要为这个结论而感到悲观。API因为需要不断改进的事实算不上什么坏事，只是对现实的一种坦诚。每一个API的作者都应该为未来的改进做出计划。这种计划是一种比较高层次的，要考虑未来版本会对API中哪些内容加以改进。这种计划可能会用到两种方式。一种极端的方式是放弃老的版本，重新开始做一套新系统。还有一种方式则是修正用户提出的问题，并强化现有的API，保证兼容性，从而使得现有客户端的功能不会有所改变。

放弃现有的API，并从头开始编写一个新的API来完成同样的任务，可以避免不兼容问题。这样做唯一的问题就在于：那些使用旧API的客户端只能继续沿用老的API，除非重新编写他们的代码，以升级到API的新版本上。所以这样做的缺点也是不容忽视的。

完全重新编写API的优点在于避免了细微的不兼容问题，但让客户端被锁定在一个特定的版本中，即使新的版本提供了大量的改进，这些客户端也无法从新版本中获益。虽然对API进行改进固然是一件重要的事情，但相比之下，兼容性更为重要。只有在这两者之间巧妙地取得平衡才能让一个API成为可用的API。

向后兼容

对于每一个API的设计者来说，都渴望做到“向后兼容”，因为不管是现在的API用户，还是潜在的API用户，都只信任那些可兼容的API。但向后兼容有多个层次上的意义，而且不同层次的向后兼容，也意味着不同的重要性和复杂度。

源代码兼容

说到兼容性，最先要面对的问题，就是保证源代码编译时的兼容。如果基于Java 1.3版本开发程序，那么可以用Java 1.4版本来编译这些程序的源代码吗？如果能做到这一点，那么可以说Java 1.3和Java 1.4这两个版本是源代码兼容的。但源代码兼容是非常难以达到的。之所以出现这种问题，主要是因为每个新版本的Java语言都会添加一些语法上面的新功能，这种改变往往都会体现在执行文件的格式上，也就是Class文件的格式会有所调整。

二进制兼容

如果一个基于老版本类库开发的程序，在不需要重新编译的前提下，可以和新版本类库进行正常连接并执行，那么这种情况可以称作二进制兼容。因为有两种场景需要这种兼容性方面的支持，所以要做到这一点也是非常重要的。首先，用户基于某个版本的类库编写了一个程序以后，原先开发的程序应该都可以一直正常运行，不管用户手中的类库是哪个版本，是否升级到了最新版本，程序的运行都应该是正常的。这样做可以极大地简化程序的维护、打包和发布工作。其次，如果用户只有一个老版本的二进制类库，也同样可以开发程序，随后再移植到新版本上，这样就无须用户来重新编译程序 。这两种场景都有各自的用途，它们提升了配置方面的灵活性，并赋予模块开发人员和用户更多的自由。为了达到这种相互调用的灵活性，开发人员至少需要了解一些源代码编译后生成的二进制格式。对于Java语言来说，就表示开发人员需要去了解Class文件的格式，以及Java虚拟机如何加载Class文件。

二进制字节码的格式与Java源代码的格式非

常相似，这有好的方面，也有坏的一面。说它好，是因为这样的格式很容易理解。说它坏，是因为它会引发一些误解。但大家应该记住，在编写API的时候，只有通过二进制格式才能最终判断不同版本的API是否兼容，也就是说代码执行时的兼容性才是最根本的。所以一定要了解Java源代码是编译成何种样子的字节码。如果有疑问的话，最好反编译一下Class文件，检查一下到底是不是自己期望的样子。有可能你会为反编译的结果大吃一惊!

功能兼容——阿米巴变形虫效应

如果一个类库在运行时，不管所引用的是老版本还是新版本的类库，其产生的结果完全相同，那么这两个版本可以称为功能兼容。这个定义看起来简单，但背后的含义却不简单。

作为开发人员，你可能会清楚地知道你所开发类库都提供了哪些功能，假设你提供了优秀的规范和完善的文档还有其他的信息，从而能够清楚地对类库的功能加以说明。当然这只是一个假设，从来都没有什么优秀的文档可以做到上面所说的目标。在现实世界中，文档总是比程序慢上一步，而且其描述的信息也只是整个程序的一部分内容。但还是先假设有一些开发人员已经完美地分析了程序，并清楚地知道程序的所有功能。如图 1中所描绘的那样。

图1　理想世界中程序功能的描述

但大家都很清楚地知道软件开发中的一条金科玉律：每个程序至少都有一个Bug。什么意思呢？所谓的Bug其实就是程序的功能不符合预期定义的内容。即使开发人员愿意相信程序的行为如图1中定义的那样完美，而现实中，程序的功能与其预期定义的内容都是存在出入的。在特定情况下，代码并没有实现预期的功能，而在其他情况下，所完成的功能要超出预期。如图2所展示的那样。

问题就隐藏在这张图的背后！假设这张图同时描述了现实世界与理想世界中的程序功能。那么圆是一个清楚的功能定义，而具体程序则有所出入，有时没有完成规范中定义的某些功能，有时却又超出了规范中定义的某些功能。而程序员在开发代码时，往往不会去阅读相应的规范。事实上，几乎没有哪个开发人员会去读这些API的规范，直到出现严重的问题。他们会用“编码/运行”这样的方式来完成自己的工作。比如说开发人员写了一些代码，运行之后发现完成了自己需要的功能。此时，他们关注的是那些API背后具体实现所完成的功能，并不关心规范怎么说。这样做，其实是让代码依赖于具体的实现，而这些实现内容并不会写到规范中。显然，不管是少提供了功能，还是多提供了功能，这两种情况都是非常危险的，会影响到未来类库版本的共存，更会影响使用这些类库的程序。一旦该类库有新版本发布，修正了某些bug，或者再添加一些功能，都有可能会影响你的程序，就像图3一样，从圆变成了不规则形状。

图2　现实世界与理想世界中程序功能的区别

图3　下一个新版本将程序变得更不规则

要为自己的行为负上责任也不是一件容易的事。所以API设计者首要的目标就是要减少阿米巴变形虫效应，要让API的功能行为尽可能地与规范保持一致。这事做起来决不简单。需要开发人员对API要完成的功能有清楚的认识，同样还要有良好的技术水平，才能将自己的意图贯彻到代码上，此外还得评估一下API的用户会如何来使用（还要想一下这些用户如何来误用API）。

面向用例的重要性

请记住，如果一个API被广泛使用了，那么就不可能了解所有使用该API的用户。比如说，Linux内核的作者不可能知道所有使用该内核的开发人员以及他们使用内核的动机所在，也不清楚地球上有多少人使用了ioctl 这个方法从内核得到相应的内容。所以，如果设计者希望能够设计出像Linux和Java这样被广泛使用的API，那么必须站在用户的角度来理解如何设计API库，以及如何才能设计出这样的API库。

既然不知道自己的客户，自然也无法进行交流，那么就有两种解决方案：一是找一些用户，对其进行研究，还是一种方式就是基于用例。基于用例也就表示站在用户的视角，然后再对部分用例进行针对性的处理。从用户处取得反馈信息，并对可用性进行研究，这是一种很好的工作方式，可以通过反馈来检验设计的用例是否正确。在做设计之前，必须进行分析，明确为什么要写这样的API，API应该长什么样，以及如何才能完成目标。

当然这些用例都是编的。当API有了真实的用户时，才会发现这些用例与与真实情况可能相

距甚远，与真实需求也有很多不同之处。从这一点上来说，第一个版本决不可能完美。但可以减少API中存在的错误。说到API设计错误，并不是指这些API不能满足用户需求，这是很正常的，因为在整个系统的生命周期中，会不断地有用户提出新的需求。所谓的错误是指API不能在后续的版本中以扩展的方式来满足用户的这些需求。但如果你学习了本书，在设计API时，不仅可以通过扩展的方式满足新需求，而且新版本也不会破坏客户基于第一个版本开发的那些代码。

前文展示的阿米巴变形虫模型说明了对外的功能描述和其内在实现之间是存在着差异的，正是这种“差异”引发了API维护中的主要问题。所以要尽可能地减少这种不一致。但想要实现这个目标也要有一个前提，就是能对外提供一个定义清楚而且明确的规范，否则没有规范，拿什么来进行比较呢！

API设计评审

过去，人们一直认为设计工作是不能由一个集体来完成的，它需要一个架构师对所有的设计进行决策。当然，这样做可以简化很多事情，但仍然有一个规模上的限制。就算不考虑模块规模大小这方面的限制，这位首席架构师的压力也是非常庞大的，其责任包括设计、维护API，还要告诉别人API应该怎么使用，这些工作内容都需要占用大量的时间，毕竟这位架构师一天只有24小时，不可能无限制地工作。

解决方法就是从团队成员中选择一些技术最好的人，指导他们来设计自己所需要的API。但这样做会造成一致性方面的问题，因为每个人在设计API时都有其个人风格。肯定无益于API的质量，必须解决这一问题。

但每一个设计良好的API，都有着相同的动机。所以要有一个团队来配合API的作者对API进行评审工作。

一旦有一个API需要改动，任何人都可以提交一个改变的请求。其他人则需要在代码正式提交前，进行一次评审，检查新调整的内容是否符合一个优秀API的基本要求。比如说，我们会按照“优秀API规则”进行检查，保证能够满足这些规则。下面详细地列出了这些规则。

用例驱动的API设计：设计API时，要基于一些具体的场景和对API的认识进行抽象分析，最终给出设计。

API设计的一致性：API往往是由多位设计者来完成的，但整个团队中必须能够保持“最佳实践”的一些基本原则。一个接口设计得再好，只要它违反整个团队的一致性，就宁愿退而求其次。

简单明了的API：简单而且常见的任务应该更容易处理。如果基于用例驱动的方式进行设计，就可以很容易地通过那些可以简单实现的场景来验证这些API是否可以完成那些重要的用例。

少即是多：一个API对外提供的功能应该只包括用例中说明的功能。这样可以避免出现需要的功能与实际提供的功能两者之间出现差异。

支持改进：以后也必须能够维护这个类库。如果出现新的需求，或者原作者离开，都不会出现放弃这个类库的情况。

一个API的生命周期

开发API的过程其实就是一个沟通交流的过程。沟通的双方就是API用户和API设计者。

API有可能是这样产生的，有些人写了一些代码，而另外的人发现这些代码的价值，就开始使用这些代码。在这种情况下，API是以一种自然的方式产生的。随后API用户和API作者有了相互沟通的渠道，开始交流经验，可能发现这个功能一开始的设计并不是很通用，或者说一开始时，作者并没有把这个功能当成一个API来设计。为了让这个功能成为一个API，他们开始讨论如何进行调整才能改善这个功能。经过几轮的迭代，才会带来一个有用而且稳定的API。

但再换一个角度来看，API设计者希望在没有对外提供一个API之前就能和相关的用户进行沟通。这种API的开发方式更接近于基于业务的设计方式。在这种场景下，系统中的两个组件间的协定是已经明确的，至少说也是已经有了需求。收集需求，定义问题域，明确用例，再由指定人员来设计API。现在，其他人员就可以使用这个API，并可以给出自己的建议，列出Bug，提出一些功能方面的改进意见。这些建议都有助于改进API，使其用途更广，也更稳定。

尽管这些API的案例各有其不同的缘由，但有相同的特点：每一个都需要时间来让用户进行试用，并进行反馈，然后才能宣称这个API是可以正常运行的。当然不是说这样做就可以带来

稳定的API，有时候，也有可能最终什么都得不到。如果出现这种情况，最好还是放弃这个API吧。有时候，可能交流的双方无法进行高质量的正式交流。在开始的时候可以简单地聊一下，交流相应的需求，但如果新发布的版本也证明了这种简单的沟通方式并不合适，那么双方也许应该更进一步地进行交流，使得沟通能更简单更有效一些。

一切皆有可能。但对于那种开发人员之间的沟通问题，最好还是要描述清楚。如果你要设计一个API，那么在这个API没有成熟前，最好能够清楚地告诉其用户：“这个API还没有完善，你可以尝试使用这个API，但一定要小心。”在有了稳定的版本以后，还可以骄傲地告诉用户：“这是我开发过的最好的API！放心使用这些API，我可以保证它能一直提供支持。”这样做可以俘获API用户的“芳心”，让他们“拜倒在你的石榴裙”下。但请一定要注意，对于API，要能够清楚地标识其当前状态，以便用户了解相应版本是否可以稳定使用。

如果想告诉类库的用户当前发布的版本还不稳定，那么最简单的方式就是把其版本标识为0.x。因为它还没有到达1.0版本，表示还在开发中，也就可能还会有所变化。不管用哪种版本标识方式，最重要的是要让API的用户清楚地知道当前版本的状态，以便他们决定如何使用该API。

逐步改善

我已经提过多次，这里再多重复一次，第一个版本远非完美。事实上，不仅第一个版本，哪个版本都不会是一个完美的版本。不管怎么样，设计的场景不可能完全准确，不可能完全符合之前的方案。对于版本间的变化，也有两种极端的处理方式：一种是逐步改善，还是一种则是完全重写。

什么叫逐步改善呢？比如说，增加了一个方法或一个类，或者是向DTD文件中加了一个新的元素，又或者是增加了一个能够影响类库功能的属性。在保证老版本API能正常运行的情况下，演化出新版本API。这种改进是一步步进行的。

关于“逐步改善”有一个谬论，就是说“因为只有少许的改动，所以以前用户使用老版本API编写的程序可以在新版本上继续运行”。每一个改变都有潜在的风险，因为它可能引发一个甚至更多的不兼容问题。每一个不兼容问题（即使看似微不足道）都会反映到客户开发的程序中，可能就会产生非常严重的后果。开始时，要能够把真正的内容与最初的设想保持一致，而且任何一个小的改变都不会引发任何问题，这样才可能避免阿米巴虫模型出现。

如果考虑到向后兼容性问题，那么也许重写一个全新的版本可能是更现实一些，这样可以清楚地告诉类库的用户，如果要移植到一个新版本上，就要花费一些时间进行代码迁移工作。与前面“逐步改善”的方式相比，这样做显得更诚实一些。但这样做也在很多方面都存在问题。首先，如果新旧版本保持兼容，那么迁移的工作量就非常小，否则完全重写一个实现需要投入大量的时间，还需要一个充分的理由来说服用户接受这样的方式。如果没有令人信服的原因来说服用户，相信用户宁愿守着老的版本。要知道，每个项目最重要的问题就是日程计划的安排。如果没有一个充分的理由，没有人愿意花费大量的时间将代码升级到新版本上，他们会去做其他更重要的事情。

如果用这种态度为API的客户提供服务，那么就不会带来一个好的合作氛围。但还有更坏的合作方式，就是完全不提供迁移的方案。有时候，对于一个愿意迁移到新版本的API客户来说，还是可以接受一个API完全重写。但如果说让所有的用户都只使用老的版本或者强迫他们立即都升级到最新版本，对于分布式开发来说，这两种方式都不现实，正如本书一开始所说的那样。如果API经常产生重大的变化，而且要强迫用户随之迁移，那么客户就会转向其他的方案，而放弃现有的API。

总而言之，还是要准备使用增量改进的方式！人们需要软件加以改进，但改进时引入的伤害也应该最小化，特别是要避免重新编写的那种大变化。如果因为API设计上的问题，使得无法增量改进，那么也许会有充足的理由进行一次重新编写，但这种大的变化应该限定于开发方式上的一些基础性变化。本书的大部分篇幅都会讨论用于API增量改进的设计实践。如果出现了很大的变化，我们会同时强调要为一个API提供多个大版本类库。只有这种方式才能保证API能够变得更好，而且使API用户的痛苦最小化。

本文摘选自《软件框架设计的艺术》（Practical API Design:Confessions of a Java Framework Architect）一书，中文版由人民邮电出版社图灵公司推出，特此感谢图灵公司的授权支持。

透明是开放平台成功的关键

——淘宝开放平台基础组件介绍

■ 文 / 岑文初

译者简介：

岑文初，淘宝开放平台主架构师。2006年加入阿里巴巴，2007年初开始负责阿里软件平台架构设计，2007年底开始进入开放平台领域，2009年8月加入淘宝开放平台。Blog: http://blog.csdn.net/cenwenchu79/。

做开放平台，在关注技术的同时，应该更多地去关注如何使平台产品化，以避免让开放成为一种空口号。本文将从开放之于服务提供者、开放之于应用开发者及开放平台体系设计中基础性的共性内容这三个方面来讲述开放平台。

做开放平台的任何公司都有自己个性化的一面，很难有简单的模式去复制、照抄。当然平台基础共性的内容是平台开放初期较为一致的内容。

开放之于服务提供者的影响和帮助

早在两年前，淘宝就开始做开放，到今年已有很多大型网站提出了开放的概念。其实每个网站发展到一定阶段时都会经历一个平台期，平台期最大的问题是如何增加用户群、如何增加用户黏度，这时就需要在原有技术和服务上做一些突破和改变。因此，平台开放便成了很多大型网站解决问题的突破口。

大多数人认为开放平台是为了创新，这没错，但创新的内涵非常丰富。创新会为突破网站的平台期提供一些帮助。大家可能已发觉淘宝做了很多方面的融合，包括与传统媒体的融合、与手机终端的融合，甚至与一些电视领域的融合，为什么呢？今天不可能把淘宝照模照样地搬到手机上、电视上、报纸上，也无法使传统制造企业上线后就直接做电子商务，因此需要开放平台来帮助淘宝实现服务的更加细化，让外部可以有更多的选择性和灵活性，以便将淘宝的血液渗透到更广泛的领域中去。

开放平台的目标就是把以往的面向技术做产品改变成面向用户做产品，让领域专家去做用户需要的产品，不再因为技术门槛而把真正懂需求的人拒之门外。同时开放也是一把双刃剑。首先，平台开放后将面临更多安全的挑战。平台的开放犹如在封闭的桶上开了一个洞，在把桶里的水放出去的同时病菌也会通过这个洞钻进来。同时开放也将面临更多的服务压力，对性能要求更高。平台开放后，其API的访问量与主站访问量的趋势、用户行为与应用行为都是完全不一样的，应用自身的稳定性对访问量也会产生影响，现在网站所能看到支撑的服务数量级，在开放后可能会相差几个数量级。

服务开放者还需要考虑开放后对已有业务的影响。以前服务提供者打算开放，直接和淘宝说一下即可开放。但现在不同了，开放之前，淘宝会询问服务提供者对这类服务的开放期望是什么？如何引导应用开发者开发服务？若自身产品与使用服务开发出的产品发生冲突后如何权衡？诸如此类问题是每个服务开放者需要考虑的，这样做是对服务开发者负责，对开放平台负责，也是对自己负责。

从架构设计来说，开放前一定要做好服务化和模块化。起初开放一定是局部性、阶段性的，不可以一次性完全开放。若没有做好服务化和模块化，就把整个系统一次性开放，可能会把整个系统搞垮，最终受伤的是服务提供者自己。

开放之于应用开发者的挑战和机遇

开发的软件与淘宝原有服务产生冲突时怎么办？如何在淘宝开发平台上赚到钱？这些都是开放平台应用开发者比较关心的问题。对于互联网应用开发者来说，产品开发和推广是最大的投入，开放平台能够给开发者提供数据来源和推广资源。另外要生存下来，开发的软件一定要有差异性，可以是行业差异、渠道差异或技术差异等。开发者不要只站在淘宝服务提供的能力去思考问题，而应该更多地结合自身的特点或者行业领域去思考问题，做一些差异化、有创新的事情，差异化可以让你避免同质化惨烈的竞争。

开放之于平台架构设计与实现的要求

在设计平台技术架构时，往往要考虑速度、稳定性、安全性、易用性等几方面。

● **速度**是服务的第一感知，淘宝已从耗带宽服务、耗时服务、耗资源服务三方面对平台技术架构做了优化。

● **稳定性**是内外兼修的工作。从系统结构设计到外部的控制台和监控系统，淘宝已对整体平台的稳定性做了大量的工作。

● **安全性**上不同的做法会使平台产品处于双输或双赢的境地，长期的双输无论对安全部门、平台产品部门还是对最终用户的使用来说，都是不满意的。

● **易用性**使应用开发者更加便利地使用平台的设计细节，而这往往是决定平台成败的关键。

透明化是平台初期设计的核心思想。透明是一切的基础，服务访问数据就是透明的分析基础，数据加上周边的一些内容，就会产生不同的效果。那么在淘宝开放平台上，当前的透明化数据是如何收集和分析的呢？这就要谈到淘宝开放平台上最基础的组件——基于统计抽象的分布式分析器。分析器和开放平台的发展一样也经历了多次进化和演变。图1是分析器的成长经历。

图1　分析器的成长经历

从图1可以看出分析器从处理的数据量到分析复杂度上都有了很大的提高，到今年年底，处理能力又提升了好几倍，同时在分析抽象模型和底层数据通信上都在不断地优化和实践。

分析器本身的架构如图2所示。

图2　分析器本身的架构

分析器和Hadoop的应用场景有所差别：首先它是抽象的统计模型，逻辑只需通过配置即可可动态加载，虽然适用面没那么广，但便于常规的统计分析；其次它适用于TB级别以下的即时分析（数据量大了也会造成网络风暴）。在设计上的差别主要在于Master与Slave之间耦合关系，分析器的Master很轻量，只关注任务简单管理（重置，接受请求然后分配）和任务执行后返回结果的合并及数据输出。Slave的工作内容全部从获得的任务中得到（任务包含了分析数据来源、分析规则和必要的分析参数等）。

图3和图4分别是基础组件协作图和系统结构图，目前还在继续改进中。

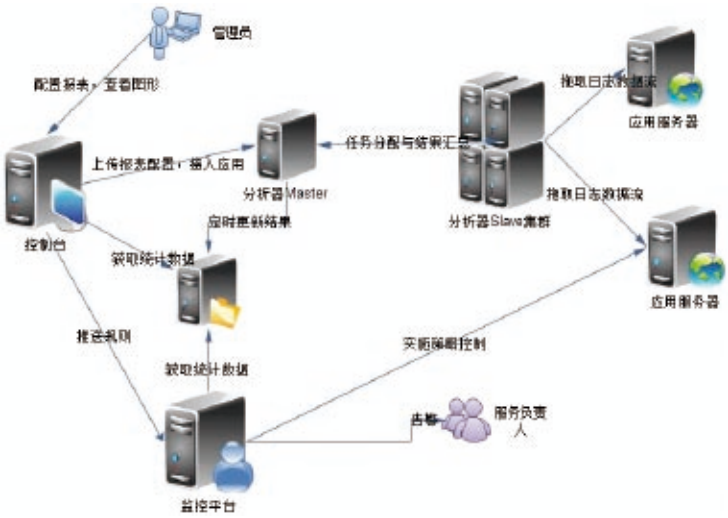


图3　基础组件协作图

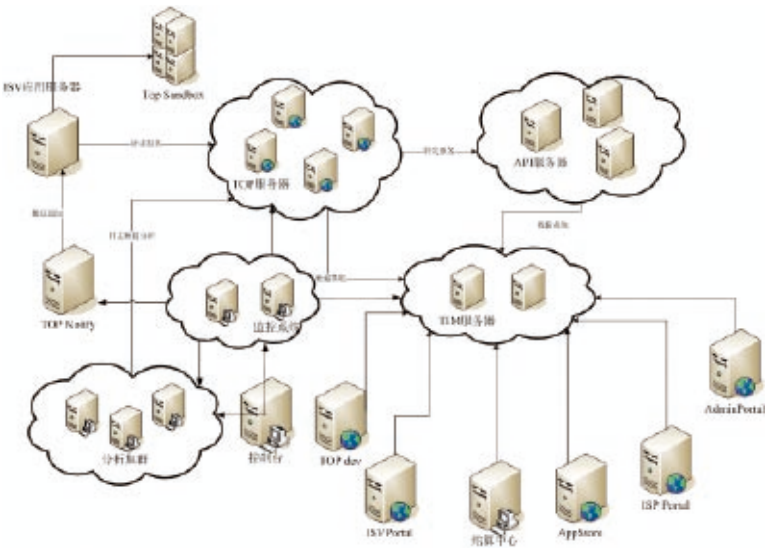


图4　系统结构图

“透明的平台是成功的平台，无声的技术是最好的技术“简单，直接”是作为程序员的我在本文最后想要告诉大家的两句话和两个词。

（本文源自作者在CSDN软件开发2.0技术大会上的演讲。）

微创业：从开放平台和云计算开始

■ 文 / 陈理捷

作者简介：

陈理捷（EasyChen），资深应用架构师，在Facebook、人人和新浪微博开放平台拥有数十个应用，累计用户数百万。2009年加入新浪研发中心，主导新浪云计算平台的战略规划和产品设计，现任SAE产品经理。

创业方式悄然更新

新浪云计算平台（Sina App Engine，以下简称SAE），SAE采用“所付即所用”的收费方式，开发者可以以很低的成本开始自己的项目。当访问量小时，一天只需要花费几分钱；当访问量大时，也只需要为自己使用的资源付费，从而不会出现因为访问量激增后又回落造成的服务器资源闲置。更吸引人的是，目前只要是能进入新浪微博应用频道的应用，都无需向SAE支付任何费用。

在架构上，SAE是分布式设计，应用从第一次部署开始就是负载均衡的，直接可用的多种服务更是可以节省大量的开发时间，只需要专注于应用的业务逻辑，将宝贵的时间和精力都用到刀刃上。

再说说新浪微博开放平台。通过它，我们可以取得数千万用户的个人信息（需要用户授权）以及用户和用户之间的关系。更重要的是，我们可以通过它，以用户身份发布微博（同样需授权），从而影响用户的粉丝，再通过他们的转发将信息进一步传递。

这使得我们能构造一个类似核裂变的信息传播链路，在短时间内覆盖大量用户，从而节省下大量的推广成本。

如何开始你的创业之路

了解开发流程

在开始开发微博应用之前，我们需要了解微博应用开发的流程，如右图所示。

在微博开放平台上，创建完应用就能获得App Key和App Secret，使用它们就可以操作接口了。但是这时候的App Key是受限制的，比如最多只能1000个用户使用；在用户授权和发布的微博来源处不会显示应用的名称。只有提交“文案审核”并通过后，才会显示名称。当通过文案审核后，应用就可以提交“频道审核”，通过审核的应用将出现在新浪微博应用频道（http://t.sina.com.cn/app）供用户选用。新通过的应用会有一段时间出现在应用频道首页的“最新应用榜”，这是我们快速获取第一批种子用户的一个重要入口。

创建新浪微博开发者帐号和应用

在成为微博开发者之前，我们必须先有一个微博账号，微博应用的基本信息和统计数据都可以在这里看到。

账户注册和应用创建部分都是常规表格，微博开放平台上有详细的图文说明。

有一个非常重要的细节是，在提交应用信息时，一定要记得划上“同时申请新浪云计算平台邀请码”，这样顺利的话，在1~2个工作日后，就能收到SAE的邀请码。

完成应用的创建后，可以在应用的管理页面看到应用的App Key和App Secret。这两个Key直接和接口调用权限关联，尤其是App Secret，不应该提供给任何第三方。

在没有邀请码的情况下，也可以注册SAE帐号，但这种体验用的帐号只能使用一周，之后将被回收。可以先创建体验帐号，在获得邀请码后，直接将其转为正式帐号。

创建SAE账号和应用

SAE是一个和GAE比较类似的平台，我们可

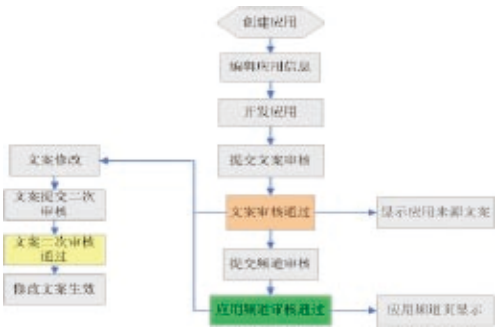


图　SAE上微博开发流程

以在上边调试、运行甚至开发Web应用（使用在线编辑器）。

和GAE不同的是，SAE支持的是在国内互联网中占主流的PHP和MySQL环境，这极大地降低了开发者的成本。一方面开发者不用再学习新的知识；另一方面有大量的现有程序稍加改动就能使用。从我们的实践来看，将WordPress移植到SAE只需要改动不到10行代码；而在SAE上使用Smarty，只需要改动两行配置。

正因如此大量的开发者选择了SAE，在微博Web应用TOP10中，有7个应用都使用了SAE，而按照目前新浪对微博应用的支持策略，这些应用不管访问量多大，开发者都不需要花一分钱。

回到用户注册上来，点击SAE首页的“创建帐号”链接，在填写开发者信息，通过邮件收到的邀请码和手机验证码后，就可以注册成为SAE的用户了。

登录后，在“应用”栏目下的“我的应用”中可以创建一个新应用。

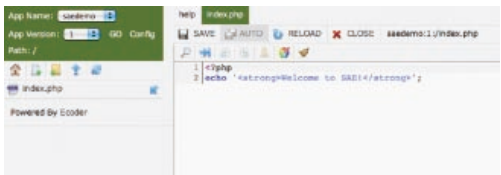
应用的Appname同时也是二级域名的一部分，所以遵循域名定义规则。创建好的应用可以用appname.sinaapp.com进行访问，目前还有一些不错的二级域名没有被注册，有兴趣的同学要抓紧哦。

开始开发应用

创建好应用后，就可以进入应用的管理界面。

从左侧的分类可以知道，这里可以查看和设置应用的基本信息、管理应用的代码和成员、在线调试和查看访问日志以及初始化和设置各项服务。

一个新的应用通常从代码管理开始。SAE上的应用支持多个子版本，可以通过“版本号。appname.sinaapp.com”进行访问。这可以帮助我们进行平滑的应用升级：先创建一个新版本，通过“新版本号。appname.sinaapp.com”进行调试，上线时只需要将新版本指定为默认版本就可以了。



点击版本后对应的“编辑代码”就可以在浏览器中使用在线编辑器（基于开源项目eCoder）编辑和发布代码。

当然我们也可以使用SAE提供的SDK来上传代码。SAE的SDK使用PHP编写，所以其命令行版本在所有支持PHP的环境下都可用。Windows用户可以选用带GUI的版本，使用起来会更加方便。

SAE平台对微博有着良好的支持，除了直接内置新浪微博的SDK外，更提供了一个可以快速

安装的微博OAuth框架。

在“应用”大分类下，选择“应用向导”中的“社会化应用”分类，就能看到“新浪微博OAuth框架”。点击进入安装，选择好应用和版本。

然后填入之前我们在微博开放平台得到的App Key和App Secret。

点击下一步，就能够得到一个可以浏览订阅和发布微博的简单微博应用。

通过修改这个已经可用的微博应用，可以快速地实现各种定制。如果想要定制一个完整的微博网站，那你可以试试“应用”分类中，“推荐应用”中的Xweibo。

注意事项

以上就是在SAE上开发微博应用的基本流程，在最后需要提醒大家注意的是，SAE的PHP环境和标准环境有一些区别。

为了保证数据一致和性能，SAE不允许对本地文件系统进行读写。

相应的，SAE提供了自己的数据存储服务Storage，其本质是用HTTP协议包装的MongoDB。数据的发布更新和访问都是通过HTTP协议，频繁的读写Storage会带来很高的资源消耗，这点在开发过程中需要特别注意。像记录调试日志这样的需求可以使用SAE_Debug函数直接输出到日志的“调试日志”分类，然后通过浏览器在线浏览，不要存放到Storage。

另外SAE提供了TMPFS服务来解决临时文件的问题。TMPFS服务为开发者提供了一个临时目录，开发者可以将文件写到这个目录下并将其路径提供给其他模块（比如cURL）使用，但需要注意的是，当请求结束时，这个文件将不再存在。

为了方便移植，SAE还可以通过PHP Wrapper用文件系统函数来操作SAE的Memcache和Storage服务。

最近SAE进行了一系列升级，已经直接支持GD、cURL（Socket还不支持）等服务，在环境上已经和标准PHP很接近。更详细的信息可以访问SAE的“开发者中心”查阅文档。

新的技术为我们带来了新的机会，祝愿各位能在这个技术主导的时代里有所成就。

“程序人生”栏目主持人

周筠，Just-pub创始人。曾带领团队策划出版了《深入浅出MFC》（第二版）、《编程之美》、《代码大全》、《程序员修炼之道》、《Unix编程艺术》等图书。关注技术，更关注技术背后的人。

我的成长

■ 文 / 吴岩峰

求学之路

1976年，我出生在一个大家庭，爷爷奶奶与父母一起生活。姐姐大我两岁，父亲是一个工程师，母亲是个普通工人。与同龄孩子不同，我在上小学前并没上过幼儿园和学前班，而是在父母的辅导下完成了学前教育。小学四年级时被选中参加一个华罗庚数学班，在一次数学竞赛中获得长春市第四名的成绩。由于成绩突出，我被保送到市里一所重点中学学习。

重点中学在教育设施和师资力量上都有很大优势，在学习基础课程的同时也办了很多类兴趣班。在上小学时我就开始接触电子游戏，所以就参加了计算机兴趣班，当初也只知道游戏是用计算机编写出来的。当第一次进入计算机实验室时，我就被它迷住了。那时的计算机与现在差别很大，机房里只有Apple II和中华学习机。中华学习机算是高档机型，可以插游戏卡。Apple II开机后只有一个内置的Basic语言，没有软盘和硬盘，也记不清是用什么来保存数据的，每次上机都要重复地输入那些Basic程序。如果在一节课上能正确地输入并运行一个小程序，就能让我激动得在实验室里尖叫。那时我还会跑到新华书店抄Basic书上的小游戏程序，然后再回来输入到计算机，但通常都是输入进去不能正常运行。初中三年很快就结束了，我也顺利升入本校的高中部。高中学习压力很大。我也不再有时间参加兴趣班。

父母在学习方面对我没有苛刻的要求，多数时候靠自觉学习。高中三年在繁忙的学习中度过。1996年我考入吉林工业大学计算机系。

充实的大学

我的大学生活非常简单，教室、图书馆、机房和寝室，四点一线。但其中很多事情对我来说又都充满了新奇。

1996年正是DOS的末期，Windows 3.1和Window95刚刚流行开来。最初上计算机实践课我们每个人都要拿着5寸或3.5寸的软盘，上课前都要自己将软盘中的DOS安装到硬盘上。机房里面多数是286计算机，好一点的是386或486。如果运气好能用上486机器，要知道，486就意味着比其他机型更快。那时DOS病毒横行，上机你要花很多时间和计算机病毒作斗争。在计算机报上经常有关于计算机病毒的介绍，出于好奇我开始关注计算机病毒，并收集计算机病毒样本。DOS时代的病毒大多都是用汇编语言写的。为了研究病毒是如何工作的，我开始自学汇编语言。DOS时代的汇编语言是一个编程利器，你可以用它去控制计算机的一切东西，包括显示图形、控制键盘、让喇叭发声，甚至控制鼠标。从那时起，我开始频繁地往返于图书馆和机房，阅读了大量汇编语言和与病毒相关的资料，我的汇编语言编程能力也突飞猛进，很快就完成第一个病毒程序。写计算机病毒也是一种编程能力和技巧的体现。从那以后有些同学开始找我帮忙调试程序和做一些课程设计。

由于经常使用DOS的Debug命令调试程序，所以我对Debug命令的实现细节很感兴趣。我开始考虑调试器的实现细节，但在图书馆里没找到关于调试器实现原理的书，最后只

有一个办法可以弄清楚Debug的实现原理，那就是通过反汇编来了解它的内部实现机理。我反汇编了整个Debug程序，并将所有代码都写在笔记本上——整整记了一厚本笔记。这也为我后期开发Syser Debugger奠定了理论基础。后来我加入了计算机中心的一个项目组，在那里我开始接触UNIX、Linux以及Windows NT，为我后期工作打下了一个良好的基础。

走出校园

大学生活很快就度过了。由于基本功扎实，我顺利地在北京一家互联网公司找到一份编程的工作。

这家公司刚成立两个月，当时只有7名员工。在那里我负责做CGI（Common Gateway Interface）程序开发。之前我并没有接触互联网的知识也不清楚CGI编程是什么概念。幸好在校期间的C和C++基础发挥了关键作用，入手轻松很多。工作是一个不断学习和进步的过程，工作的压力使学习和进步的节奏加快。由于工作需要，我开始接触和使用其他编程语言，如PHP、ASP、JSP等更高效的CGI开发语言，在提高工作效率的同时也使公司的业务量不断扩大。公司在一年内也扩大到50个人的规模。这些脚本语言开发更加方便快捷，并且不需要编译和链接。毕竟这些语言是专门为写CGI程序而开发的，它们有更高的开发效率。在最初要换这些编程语言时，同事们都有比较抵触。这种抵触情绪是每个程序员发自潜意识的，就像微软的Windows NT核心团队抵触C++语言一样，都拿代码运行效率来说事。但当用了这些脚本语言一段时间后，我们发现它们的优点完全可以弥补C和C++带来的损失，况且需要执行效率的地方还可以使用C和C++。

这段工作经历使我意识到接受新事物以及通过不断学习更新知识结构的重要性。随着第一波互联网泡沫的破灭以及公司规模的迅速膨胀，公司在资金上遇到了大麻烦，面临解散或者调整业务方向以并入投资人的其他公司这两个选择。最后公司被并购。在寻找新的发展机会和留在这里继续工作中，我选择了寻找新的发展机会。三年工作使我在技术方面对自己更有信心。此时我已不再是刚毕业的毛头小子，虽然很喜欢互联网公司，但内心还是希望找到一份自己更喜欢的事。尽管这三年没有再接触计算机病毒技术，但它依然对我有巨大的吸引力。与互联网公司相比，反病毒公司是一个传统意义上的软件公司，在工作模式上有很大差别。于是一个很现实的问题摆在我面前：是否放弃三年的工作经验重新选择一家差异很大的公司？最终还是兴趣起了主导作用，我加入了瑞星。

动力源于兴趣，加入瑞星两个月后我就掌握了大部分的病毒技术。由于工作突出，我被提升为病毒组组长。于是工作中多了一些管理任务，也需要配合市场或宣传部门做一些采访和拍摄任务，偶尔还需要去面对客服人员的问题。这期间让我学到不同团队间合作的重要性。不过在这个职位上编程的工作非常少，都是做二进制代码分析。病毒分析是一个有趣的工作，在这个过程中你可以了解病毒作者的编程思路和编码技巧，这也是一个学习的快速途径。在分析病毒的过程中可以接触到Windows的各个方面，包括应用程序、API、驱动、内核等知识。

在病毒组的8个月，我长时间分析他人的程序，不亲自写程序，这使我意识到自己的编程能力在下降。这是我不希望发生的事，我更喜欢编程序。通过争取我加入到公司的新技术组，这是一个研究性质的部门，主要任务是研究反病毒技术的未来发展方向，以及如何把新技术加入到产品中去。这个部门里都是技术专家，陈俊豪就是这里的一员。他是一个极富编程天赋的小伙子，在技术方面我们有很多共同语言。调试器是我们必不可少的工具。我们在使用现有调试工具过程中发现了某些不足，Syser Debugger的一个开发计划就在这样的需求下诞生了。在2~3个月的时间内，Syser Debugger的原型被开发出来，但它是如

作者目前从事便携设备操作系统软件开发工作

此简陋以至于只有2~3个命令可以使用。软件是一个循序渐进的过程，是从不完善到健全、从简陋到充实的过程。在开发过程中，我们遇到过很多技术难题，需要不断地去解决各种问题。

有一个印象深刻的Bug，只发生在多核CPU上，并且还与CPU的速度有关，在慢速CPU上它很难重现。它会随机导致操作系统宕机。为了定位它，我每天下班回家都要调试两个多小时。就这样坚持了两个月，期间几次我都要放弃，有一次还把键盘砸个粉碎。最终通过排除法，几乎把所有的代码都注释掉，才发现了这个问题。问题是多核CPU导致中断重入，破坏系统堆栈。而这个问题的解决只需要调换两条汇编指令的顺序。但当有用户反馈你的工具能帮他们解决问题时，你就会很激动，重新燃起工作的热情。Syser Debugger的开发占用了我们所有的业余时间。我想Syser Debugger能够成为一个产品，最主要的原因是持之以恒的改进和完善。做任何事情都需要这种精神。在瑞星的三年多时间是我技术进步最快的阶段，也学到了如何管理技术团队。在这里我认识了很多优秀的技术人才，也非常感谢公司能为我们创造一个很好的工作环境。

在瑞星的后期，由于网络游戏快速发展，也带动了游戏外挂这个黑色产业的发展。我的经理也加入这一新兴产业，居然利用公司资源为自己开发游戏外挂。他的这一行为，产生很多连带的负面效应。我婉拒了经理的邀请，没有加入这个行业。后来他得到了法律的制裁。回想当初，如果我加入的话，那将是一场噩梦。由于公司环境的变化，我最终选择离开。

由于外资IT企业有先进的软件开发管理流程，以及完善的软件质量保证体系，所以我打算进一家外资软件开发企业去学习和感受它们的先进管理方法。进入外企首先要解决英语这个拦路虎，而我的英语可以用糟糕来形容，学了那么多年，仅仅认识一些单词、能阅读技术文章而已。通过两个月的外语强化学习，最终我被一家美国的网络安全软件公司Websense录用。由于公司产品针对企业级用户，所以质量方面有严格的要求，基本上一个开发人员会对应一个测试人员，这在国内很难做到。很多外资企业都是多区域办公，软件产品也是跨区域开发，这就对管理有更高的要求。软件开发过程也被分割成很多细小的单元，基本上是按每人每周为单位作为检测点，甚至是更小的单位。工作的细化能够确保一个大的工程能按计划完成，让员工有强烈紧迫感。这也需要管理者有能力去细化工作，从而促进工作岗位的细分。

跨区域开发中，文档是大家交流的基础，对文档的要求相当苛刻。一个开发人员往往需要花费一半甚至更多的时间去写文档，其余的时间才是真正的开发工作。测试人员是为文档服务的，没有文档测试人员是不会开始工作的。文档也是知识积累的最佳途径之一。我之前工作的公司都不重视文档，很多知识都是保存在个人大脑中，人员的流失也会导致公司技术知识的流失。这也恰恰是很多软件公司很难发展壮大的原因之一。公司也很重视公司影响力的宣传，有专门的岗位负责公司Twitter、Blog发布，在员工培训方面也有专门部门负责。

当然外企也有自身的缺点。在国内分部很难接触到公司的核心项目开发，往往是做一些边边角角的工作或总部不喜欢做的繁重体力活。这对个人技术发展不利，尤其是对刚刚参加工作的员工。分工的细化也导致员工的技能方面得不到全面的发展。外国人有种天生的优越感，有极少数人甚至到了傲慢的地步。我还记得有一个法国人来北京开会，直接对在会议室的中国员工说，你们有人懂UNIX吗？你们就没有人懂UNIX。这简直就是侮辱。由于管理的繁杂，也导致了效率的低下。日常工作都是以邮件为基础展开，由于跨越不同时区，也导致交流的时间成本过高。通常一个简单的软件Bug，花在交流上的成本就要1~2天。外企和国内公司相比更制度化些，尤其在经济危机时期，裁员行动相当迅速敏捷。总的来说，外企在文档和测试的投入方面比国内公司大，这也确保产品有一个长远的发展。这点是国内公司应该学习的地方，很多公司做事情都是急功近利、急于求成。

作者简介 | 吴岩峰

盛大创新院资深研究员，对软件调试有深入了解。现致力于便携设别操作系统软件开发。

■ 责任编辑：董世晓（dongsx@csdn.net）

Facebook工程管理揭秘之一

招聘是第一位的

■ 文 / 黄易山

Facebook前工程总监黄易山（Yishan Wong）撰写了一系列文章，很好地总结了Facebook卓越研发文化中的宝贵经验。本刊将陆续连载这一系列，本文是第一篇。

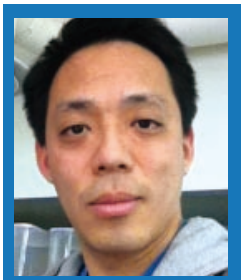
从2006年底到2009年初，我有幸在Facebook的工程部门先后担任了不同的管理职务，包括几个不同团队的经理，以及工程总监，也见证了工程部由约30个人发展到200人左右。这段时间基本上跨越了从动态消息功能（News Feed）、Facebook平台（Facebook Platform）在第一届F8大会上的发布，到自助式广告系统（现在是我们现金流的主要来源）、网站国际化以及Facebook连接（Facebook Connect）的推出。2006年，我们还只是一家用户数不足1000万的基于大学的社交网络，而到了2009年初，我们在全球已经拥有超过2.5亿用户。这期间，公司也从一家小的创业公司（拥有不到100名员工）成长为一家中等规模的公司（拥有800多名员工）。

我到Facebook工作时，公司已经很明显将会迅速成长。我当时怀着这样的宏愿：在这个决定性的时刻，我要成为影响其关键决策的一份子，使在遥远的未来，Facebook和其工程部门能成为一个充满活力和具有持续性的机构。从我在其他经历了这种高速增长期的公司工作的经验中，我总结了一些有助于建立一个伟大工程环境的关键文化因素和组织因素。一方面，优秀人才愿意在这种环境中工作；另一方面，该环境允许最大限度地创新及快速地执行。如今，我又回到了一线工程师岗位。某日，我反思自己为何会享受在这样高效的工程环境下工作、为何会如此乐在其中。同事给我的答案是：“这就是建立优秀工程部门的‘Yishan’秘诀？”我从未想过以如此教条的方式来表达我的想法（实际上，这样做是危险的，因为它们会被神化），但无论如何，我决定努力试试看，看我能否将它们列成下面这样的清单。

- 招聘是第一位的
- 让亲身实践者执行工作流程
- 内部晋升
- 工欲善其事，必先利其器
- 技术型领导

注意，这其中并没有如何建立技术创业公司的各种显而易见的硅谷经验，例如“雇佣最优秀的人才”或者“建立一个可以确保开放沟通的环境”，因为这些早已众所周知了。清单中列出的是我认为不那么明显的事情，而这些事情很容易被快速增长的技术组织所忽视。相信那些能成功地将它们融入文化和惯例的公司，最终将变得更强大、持久、自新，而那些没这么做的公司，最终会变弱、陷入平庸。

在接下来，我会就每一点写一篇文章，分别阐述它们的含义及其之所以重要的原因。那些在过去几年里与我共事的人们，如果你看到这些文章，就能理解我当时为什么这么做了。希望大家能感受它们的用处和乐趣！



黄易山

1997年毕业于卡内基-梅隆大学。2001年加入PayPal，曾任高级工程总监。2005-2010年在Facebook领导研发，在公司研发环境的建设上发挥了重要作用。

招聘是第一位的

招聘是第一位的，也就是说，“永远将招聘作为你的第一要务”：要将招聘作为你所在部门的第一要务、作为每个经理的第一要务、作为每个工程师的第一要务。

招聘需要多个部门的合作，并且包含很多阶段。概括起来就是：寻找候选人、筛选、面试、作决定、发出邀约、结束招聘。

在每个阶段里，要将与招聘有关的行动作为责任人的首要任务。举例来说，招聘人员需要及时与负责人进行沟通，并且在人力所及的范围内尽快安排面试以及后续的谈话。也就是说面试要优先于其他一切工作。要根据面试的结果尽快作出聘用或者不聘用的决定，并且尽快地发出工作邀约。如果能今天打给应聘者，招聘人员就不该等到明天；如果能安排这周面试，就不该等到下周。面试官不能因为其他事情推掉面试，也不能在几个小时甚至几天后才对面试做出反馈。招聘经理将反馈意见汇总到一起，以决定是否聘用，组织出一份邀约，并迅速结束对这名候选人的招聘。

各个部门是否赞成“招聘是第一位的”这种价值观，必然会对招聘过程的执行效率带来一些副产品和积极的因素。在聘用某些候选人（通常来说，是满足职位要求的候选人）时，有意识地“将招聘作为你的第一要务”正好提供了一种竞争优势；但它真正的好处在于促进了其他行为、价值观，结果有效地提高了员工的素质和工作场所的品质。

影响

- 引起了对高素质人才招聘标准以及如何在实践中施行此标准的重视。“雇佣最优秀的人才”是硅谷的一条至理名言，但很多公司却并没有能区分出“雇佣最优秀的人才”和“雇佣你所面试的候选人中最优秀的人才”之间的差别。成功地雇佣到“最优秀的人才”是你将招聘作为第一要务所带来的副产品，因为要将最优秀的人才吸引到你的公司来，需要将他们列入可选的名单，再对他们进行有效地评估，然后从竞争激烈的邀约中脱颖而出——要达成这样的整体效果，需要每位相关人员全心投入，并且做好每一件基本的工作。

如何才能设定高标准？如何测试一个候选人是否符合这些标准？这两件事都是说起来容易做起来难，因此只有让招聘在大家的意识中占据首要位置，普通员工和经理们才能自然而然地将精力放在诸如此类的事情上：哪些是有效的面试技巧，面试中问哪类问题最好，如何有效区分面试中的好征兆和坏征兆，以及训练面试人员中的骨干后续如何有效、重复地实践以上技巧。要自发地做到这些，需要耗费巨大的集体精力，而且除非每个人都沉

浸在一种“招聘压倒一切”的文化氛围中，否则这一切都不可能发生。招聘是件混乱、无法精密计划的麻烦事儿（这也正是技术人员所痛恨的），所以除非招聘人员确信它很重要，否则你没法调动他们的积极性。

● 这也意味着每个人都是面试官（或者说公司里的每个人至少有一部分时间是面试官）。既然招聘是每个人的当务之急，那么没有哪个人可以避免参与到面试（以及其他与招聘有关的活动）中来。应该避免这样的情况——某个工程师说：“我只想写代码，你们这些人去做面试和招聘工作吧。”

这会造成一系列影响。首先，它意味着更多的人会和最终加入的候选人接触，这有助于团队联系的快速建立——新人进入工作场所时，而其中有一些人是他已经见过面的，并且他知道这些人赞成并欢迎他的到来。其次，它把维持工作场所品质的权力交到了每个人的手中。人员的高素质不只是管理人员在象牙塔里决定的事情，它与每个人的行为都息息相关。在高速增长的公司中，最常问到的一个问题是：“我们怎样确保雇佣到的人是最顶尖的？”答案是：如果每个人都参与到面试中来，都以此为己任。公司里的每个人都应该承担以下的责任：不断使自己更善于判断某个人是否有资格加入公司；对候选人进行严格的测试；在不确定该候选人合格时说“不”；而最重要的也许是，坚定立场并且坚持否决招聘（“这绝对不行，除非我死了”），这是因为你在某个候选人身上看到了无法忽略的危险信号，因为你也是标准和公司文化的捍卫者。

同事的素质是工作场所幸福度的最大决定因素，而改善大家工作体验的最主要的方式就是，每个人都充分地参与其中。

● 它也改善了你寻找候选人的渠道。招聘人员无法搜寻到和筛选出最佳的技术人才，因为他们本身不是技术人员。以招聘作为优先任务会激励你现有的最优秀的员工，他们会寻找和推荐接触到的最优秀的人选，这就构成了你成功的候选人渠道的主体。同样，这并不容易做到（一个天然的障碍是，人们不愿去一遍又一遍地烦扰自己有才华的朋友），除非大家认为这件事是当务之急。

● 你会得到（客观上的）最优秀的人选。这是引人注目的、理想的最终结果，但是只有在给予足够的重视和行动后，才后出现这种结果。招聘是一个零和博弈游戏。候选人不加入你的公司，就会加入你的竞争对手，你的损失将是他们的收益。如果没有将招聘作为首要任务，就不太可能会在招聘中领先。这意味着别人会领先，真正的最佳候选人将会加入他们，而你雇佣到的是“能面试到的最优秀的人选”。

长期影响

● 最明显的一个长期影响是，对招聘的重视造成的影响反过来促进了这种重视。一旦这种重视能带来成功的招聘，就会很容易延续下去。特别是优秀的人才加入公司，并且积极地支持这种文化氛围，以确保更多的优秀人才加入。这些影响要在几年后才会显现，因为即使拥有比竞争对手略微优秀的人才，随着时间的推移，也会形成倍增的效果。首先，你努力聘请到最优秀的人才，然后你得到最优秀的人才，然后你得到最好的结果。

要将招聘作为你所在部门的第一要务、作为每个经理的第一要务、作为每个工程师的第一要务。

● 在更广泛的组织水平上，这也会产生高素质的招聘经理和经理候选人（那些日后可以从内部提拔为经理的人）。优秀的管理者会认识到人的重要性，并且寻找愿意将这种价值观应用到实践中的组织。拥有高素质的管理人员，对公司运营的其他几个重要方面（包括有效的绩效管理，设计和部署有效的激励机制，推动有强大执行力的企业文化）是至关重要的。

● 成功地雇佣各个级别的优秀人才，意味着一段时间以后，你将拥有强大的内部晋升渠道。这使你可以更容易从内部提拔人员和减少寻找外部候选人（这总是存在风险的）的必要。反过来，这意味着新的主动权、组织的成长、战略上的变化、对退休领导人的补充——任何可能需要新领导者的情况——都可以搞定，而且能够有更优的人选，风险也更小。

■ 责任编辑：董世晓（dongsx@csdn.net）

让Spring更敏捷

■ 文 / 何坤

SpringX框架将“约定优于配置（CoC）”的思想做了最大发挥，实现了业务bean零配置及按需加载、敏捷单元测试和敏捷AOP配置等三大特性，能大幅简化Spring的配置，显著提高应用和单元测试执行速度，实现Agile Spring开发。

Spring在企业和电子商务网站应用普遍，在使用中都遇到哪些问题和困扰？以下是我所遇到的几个问题。

配置管理问题：初次配置虽不麻烦，但业务bean配置量太大；bean归类后，配置文件过多，配置管理、查找和维护颇有代价。

应用启动速度很慢：程序员启动WebApp，有时只为测试自己开发的模块，而系统会加载所有bean的定义，启动速度很慢；即便启用延迟初始化（LazyInit），效果也不明显。

单元测试速度慢：在Eclipse内，程序员只想快速运行当前的UnitTest类，而Spring却需要20秒时间启动，因为它需要加载所有bean定义。

Spring AOP的配置烦琐：为一个bean进行方法拦截，需要配置多个AOP组件。

带着这几个问题，我们通过分析Spring的设计目标和原理来探索一下更敏捷的解决方案。

Spring设计目标和原理

Spring框架的核心设计理念是控制反转（Inversion of Control，IoC）和依赖注入（Dependency Injection，DI），通过XML配置文件实践bean组件之间的依赖装配。

如图1所示，目前基于Spring思想的企业开发模式是Action + Service + DAO三层架构。

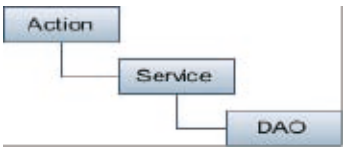


图1 Spring 企业开发三层架构模式

例如对于Order管理用例，基本业务组件和代码文件有以下三个部分：

组件类型	组件名称	功能说明
Action	OrderAction	表示层控制器，封装表示层逻辑
Service	OrderService	订单服务，封装可复用的业务功能
DAO	OrderDAO	订单数据访问对象，封装订单相关的基本数据操作。

其中依赖注入都需要一个setXxx(XXX xxx)方法，例如：

```
package org.bamboo.springx.test.demo.service;
public class OrderService {
    private OrderDAO orderDAO;
    public void setOrderDAO(OrderDAO orderDAO) {
        this.orderDAO = orderDAO;
    }
}
```

该用例的Spring配置通常为applicationContext.xml，订单管理用例配置如下：

```
<bean id="orderAction" class="org.bamboo.
springx.test.demo.action.OrderAction">
    <property name="orderService" ref=
    "orderService"/>
</bean>
<bean id="orderService" class="org.bamboo.
springx.test.demo.service.OrderService">
    <property name="orderDAO" ref=
    "orderDAO"/>
</bean>
<bean id="orderDAO" class="org.bamboo.
springx.test.demo.dao.OrderDAO"/>
```

Order功能的业务bean配置7行，如果启用byName或byType装配，可以缩减到3行。

一个大型系统，如电子商务网站、移动BOSS系统等，由大量用例组成，并且随着应用的发展，用例数也会正比增长。体现在Spring 配置上，就是XML配置量很大、bean数量迅速膨胀。

这种模式的问题是：业务bean的新增、修改、测试需要花费时间；应用后期维护期间，大量的bean管理难度加大。例如花费时间去查找某个Action和诸多Service之间的依赖注入关系、去除某个bean时需要周密排查和测试等。总之，这些都显得非常不敏捷。

按约定编程

一个大型应用中，Spring 配置主要包括两类：环境bean配置和业务bean配置。

环境bean配置特点是外部提供参数，并且变化频繁，例如DataSource、Hibernate配置、JMS等；业务bean配置主要是三层架构下的Action + Service + DAO，定义业务组件的bean id，实现类和依赖注入关系。对于业务bean配置，其package命名、class命名、setter方法命名等可以遵循一些约定的规范。

Spring组件扫描机制

Spring的基本原理是解析XML中的<bean/>定义，来管理bean，藉此实现IoC、DI和Autowire等机制。其基本过程如图2所示。

图2　Spring运作原理

针对大量的业务bean配置，如果遵守统一命名规则（CoC思想），问题将变得比较简单。Spring从2.0开始提供了组件扫描机制（component-scan）来解决这个问题，使用方式如下：

```
<!-- spring业务bean零配总控 -->
   <context:component-scan base-package
   ="org.bamboo">
   <context:include-filter type="regex"
   expression="org\.bamboo\..*\.web\.
   action\..*Action"/>
   <context:include-filter type="regex"
    expression="org\.bamboo\..*\.
   service\..*ServiceImpl"/>
   <context:include-filter type="regex"
    expression="org\.bamboo\..*\.
   dao\..*DAO"/>
   </context:component-scan>
```

component-scan的基本原理是对classpath中的所有class进行扫描，将匹配过滤器include-filter正则表达式的类自动注册为BeanDefinition。因此可以大幅减少业务bean配置。

但是组件扫描机制也存在缺点：全量加载。include-filter会扫描并加载JVM中所有匹配规则的类。而在大型应用中，业务bean的数量多，有多少个DAO类匹配规则，都会被加载及初始化。因此，全量加载和初始化会导致Web Application和UnitTest启动过慢的问题，导致开发过程不敏捷。

在大型应用中，经常遇到不需要的组件也被扫描进来，导致错误地自动装配（Autowire），Spring的机制显得比较机械。因此，在大型系统中一般不敢轻易使用组件扫描机制。

SpringX 的业务Bean零配置

经以上分析，我们的目标是既能简化Spring配置，又能实现按需加载、快速启动Spring容器，提高开发效率。

SpringX的零配置特性

SpringX基于按约定编码（Code by Convention）思想，提供“动态自动装配器”、“接口实现映射器”等组件，实现业务bean的零配置。

业务bean零配置

与Spring标准配置方式相比，SpringX使applicationContext.xml不需要像前面提到的7行或3行配置。一个有几百个用例的应用，可以不做任何业务bean配置，配置就会大幅减少。

按需加载，动态装配

SpringX的这个特性使得Web Application和UnitTest启动更快速，实现了最初的Spring敏捷化的目的。SpringX如何实现这两个敏捷特性呢？与Spring的组件扫描机制有何不同？

基本用法

下面我们来看在Spring的标准用法中，初始化容器和获取bean是如何实现的。

首先在applicationContext.xml中用<bean id=“”/>标签配置orderAction、orderService、orderDAO等组件。使用代码如下：

```
ApplicationContext context = new ClassPat
hXmlApplicationContext("applicationContex
t.xml");
OrderAction orderAction = (OrderAction)
context.getBean("orderAction");
orderAction.createOrder("AA001");
```

而在SpringX中，首先，<bean/>标签配置不需要。用法为：

```
ApplicationContext context = new ClassPat
hXmlApplicationContext("applicationContex
t.xml");
BeanFactory beanFactory= BeanFactory.wrapA
pplicationContext(applicationContext);
OrderAction orderAction = (OrderAction)
beanFactory.createBean(OrderAction.
class);
orderAction .createOrder("AA001");
```

其中BeanFactory.wrapApplicationContext方法用来将Spring的applicationContext包装为一个SpringX的beanFactory。而获取bean实例的方法，由 context.getBean(“orderAction”) 改为beanFactory.createBean(OrderAction.class)，它的参数是class而不是String类型的beanId。

一般来说，Spring和SpringX是底层框架，业务代码不需要直接使用这些操作，而由应用级框架来操作，例如Struts等MVC框架。即便业务代码使用beanFactory.createBean(class)方式，它的强类型依赖特性也由于context.getBean(string)，代码健壮性更强。

SpringX原理1——依赖检查

SpringX提供了一种依赖检查（Dependency Check）的技术，来分析被获取的bean的属性，通过属性的setter方法的参数类型签名判断出依赖的业务bean的类型Class。

例如，找到OrderService的setOrderDAO(OrderDAO orderDAO)方法，就可以判断出，该属性依赖的bean类型是org.bamboo.springx.test.demo.dao.OrderDAO。依赖检查是递归的，从Action→Service→DAO链条上的所有需要的bean都能被按需注册和装配。这与Spring“组件扫描”技术采用了完全不同的思路，也是实现“按需加载”机制的关键。

SpringX的核心架构如图3所示。

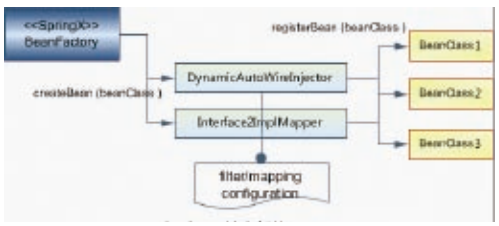


图3 SpringX的核心架构

SpringX原理2——动态bean注册

SpringX基于Spring核心封装了一种动态bean注册技术，通过“beanFactory.createBean(OrderService.class);”方法可以在无配置的情况下，根据class类名动态创建bean定义，并注册到当前的spring context中，就像添加了以下一段配置一样：

```
<bean id=”orderDAO”  class=”org.bamboo.springx.test.demo.dao.OrderDAO” />
```

SpringX核心组件1——动态装配器

对于OrderAction的setter方法，并非所有方法都是用来进行依赖注入业务bean的，例如以下方法：

```
setOrderService(OrderService orderService)
setContext(Context context)
setDataSource(DataSource ds)
```

其中，后两个setter方法是环境变量，无需动态装配。为此，SpringX 框架提供了动态装配器（DynamicAutoWireInjector）组件。

动态装配器的职责就是，通过自动装配过滤器（autowireFilters）来判断需要动态注册和装配的依赖bean。它定义如下：

```
<!-- 依赖属性为 业务 *Service 可注入 -->
<!-- 依赖属性为 业务 *DAO 可注入 -->
<bean id=”org.bamboo.springx.DynamicAutoWireInjector” class=”org.bamboo.springx.DynamicAutoWireInjector”>
    <property name=”autowireFilters”>
        <list>
            <value>org\.bamboo\.springx\.test\..*\.service\..*Service</value>
            <value>org\.bamboo\.springx\.test\..*\.dao\..*DAO</value>
        </list>
    </property>
</bean>
```

上述配置表明，容器中只有class name符合此规则的Service和DAO类才需要按上述规则处理。

SpringX核心组件2——接口实现映射器

通常在业务系统中，服务组件Service是以“接口+实现”的方式设计的。例如：OrderService是接口（interface）类型，无法实例化，因此无法注册bean定义，那Spring如何初始化它呢?

这就需要用到另一个组件：接口实现映射器Interface2ImplMapper。它的职责就是告诉SpringX遇到接口类型时，如何去找它的bean实现类。

```
<bean id="org.bamboo.springx.Interface2ImplMapper" class="org.bamboo.springx.Interface2ImplMapper">
  <property name="interfaceClassMapping">
    <props>
      <prop key="org\.bamboo\..*\.service\..*Service">
        org\.bamboo\..*\.service\..*ServiceImpl
      </prop>
    </props>
  </property>
</bean>
```

以上配置表示，服务接口名加“Impl”后缀可以判断出Service的实现类名。

SpringX核心思想

按约定编码（Code by Convention）

与Spring的组件扫描机制类似，DynamicAutoWireInjector组件通过正则表达式org\.bamboo\.springx\.test\..*\.service\..*Service来代表一类组件。因此核心思想是按约定编码。

例外配置（Config by Exception）

凡事都有例外。当OrderAction依赖的orderService是一个外部实例，例如一个远程SCA服务bean：com.xyz.saling.service.orderServiceImpl时，它并不匹配DynamicAutoWireInjector的名称规则，那么就需要例外配置一个<bean/>：

```
<bean id="orderService" class="com.xyz.saling.service.orderServiceImpl"/>
```

SpringX在bean注册阶段，能自动发现id为orderService的订单服务已经在XML被注册，因此会按byName方式将其注入给orderAction，也就是说XML的bean定义优先。

SpringX充分遵循了“按例外配置”（Config by Exception）的思想。因此，在保证大部分bean配置敏捷化的同时，又不失灵活性。

SpringX 敏捷Unit Test

在网站项目中，程序员在Eclipse里运行单元测试一直有个苦恼：只想快速运行一个OrderActionTest类以检查代码是否符合预期，而Spring配置了几百个的Action、Service、DAO，于是不得不全部加载定义和初始化，导致容器启动很慢，影响单元测试的开发效率。

而SpringX不同，它允许你不配置任何业务bean。而运行OrderActionTest时，只需要注册与订单功能相关的3个bean：OrderAction、OrderService和OrderDAO。容器启动速度和单元测试执行的速度大大加快。这也是 Spring的组件扫描机制无法满足的。

以订单Action单元测试为例，SpringX的UnitTest用法如下：

```
import org.bamboo.springx.test.DbUnitPreparedDatabaseTransactionalTests;
public class OrderActionTest  extends DbUnitPreparedDatabaseTransactionalTests {
    protected OrderAction OrderAction;
     protected void prepareTestInstance()
          throws Exception {
              super.prepareTestInstance();
              OrderAction = (OrderAction)
              getBeanFactory().
              createBean(OrderAction.class);
          }
          public void testupdateOrder() {
             Order orderInSession = orderAction.
             updateOrder(orderItems,
             orderInSession);
             assertNotNull("orderInSession",
             orderInSession);
             //other assert
}
```

其中DbUnitPreparedDatabaseTransactionalTests基类参考了Spring Test子框架，实现了同样的DBUnit和事务回滚支持。createBean(OrderAction.class)方法基于SpringX完成bean的依赖注册和注入工作。

SpringX敏捷AOP

Spring的AOP配置

AOP是Spring 重要特性之一，运用广泛。下面以日志切面拦截器（LogAdvisor）为例，介绍Spring AOP的基本配置方式模式。

业务需求

对DAO类的create、update方法实现业务日志、性能分析、性能监控等功能。

要求不用AspectJ，不用任何AOP Annotation侵入Java代码，不使用base bean继承方式。

设计实现

给定三个拦截器logAdvisor、performanceAdvisor、monitorAdvisor，需要通过Spring提供的表1所示的组件为DAO组件做拦截配置。

```
<!-- 给定3个功能拦截器：日志、性能、监控-->
<bean id="logAdvisor" class="org.bamboo.springx.log.LogAdvisor"/>
<bean id="performanceAdvisor" class="org.bamboo.springx.PerformanceAdvisor"/>
<bean id="monitorAdvisor" class="org.bamboo.springx.MonitorAdvisor"/>
<!-- BeanNameAutoProxyCreator，匹配要拦截的bean名称 -->
<bean id="DAOAutoProxyCreator" class="org.springframework.aop.framework.autoproxy.BeanNameAutoProxyCreator">
    <property name="proxyTargetClass"
    value="true"></property>
    <property name="beanNames">
        <list>
            <value>*DAO</value>
        </list>
```

表1　Spring提供的用做拦截设置的DAO组件

Spriong AOP组件	组件名	功能
BeanName自动代码生成器	BeanNameAutoProxyCreator	按bean-Name匹配
Method匹配通知器	NameMatchMethod-PointcutAdvisor	按method-Name匹配

```
    </property>
    <property name=" interceptorNames" >
        <list>
            <value>logMethodPointcutAdvisor
            </value>
            <value>performMethodPointcut-
            Advisor</value>
            <value>monitorMethodPointcut-
            Advisor</value>
        </list>
    </property>
</bean>
<!-- 1. 日志方法拦截器  -->
<bean id="logMethodPointcutAdvisor"
class="org.springframework.aop.support.Na
meMatchMethodPointcutAdvisor">
    <property name="mappedNames">
        <list>
            <value>create*</value>
            <value>update*</value>
       </list>
    </property>
    <property name="advice">
        <ref local="logAdvisor"/>
    </property>
</bean>
<!-- 2. 性能方法拦截器  -->
<bean id="performMethodPointcutAdvisor"
class="org.springframework.aop.support.Na
meMatchMethodPointcutAdvisor">
    <property name="mappedNames">
       <list>
            <value>create*</value>
            <value>update*</value>
       </list>
    </property>
    <property name="advice">
       <ref local="performanceAdvisor"/>
    </property>
</bean>
<!-- 3. 监控方法拦截器  -->
<bean id="monitorMethodPointcutAdvisor"
class="org.springframework.aop.support.Na
meMatchMethodPointcutAdvisor">
    <property name="mappedNames">
       <list>
            <value>create*</value>
            <value>update*</value>
       </list>
    </property>
    <property name="advice">
       <ref local="monitorAdvisor"/>
    </property>
</bean>
<!-- 4.  其他方法拦截器  -->
```

我们观察到，为了给DAO增加多个切面功能，它需要1个beanName 代理创建器+ 3个Method切面通知器，配置比较烦琐。

出现以上问题的原因是Spring的方法切面通知器（*MethodPointcutAdvisor）的advice属性一次只能引用一个拦截器。因此每增加一个对Method的切面过滤，就需要增加一个方法切面通知器。当需要增加更多方法切面需求时，势必需要配置更多。

SpringX的AOP配置

从上面的配置可见，bean筛选和method筛选如果能放到一起，配置将会大大简化和易读。

SpringX通过SimpleNameMatchAutoProxyCreator组件将以上两个AOP bean进行封装，同时对bean name和method name进行匹配。简化后配置如下：

```
<bean id="DAOSimpleNameMatchAutoProxyCreato
r" class="org.bamboo.springx.aop.autoproxy.
SimpleNameMatchAutoProxyCreator">
    <property name="proxyTargetClass"
    value="true"></property>
    <property name="beanNameAndMethodNames">
        <list>
            <value>*DAO,delete*</value>
            <value>*DAO,update*</value>
        </list>
    </property>
    <property name="interceptorNames">
        <list>
            <value>logAdvisor</value>
            <value>performanceAdvisor</value>
            <value>monitorAdvisor</value>
        </list>
    </property>
</bean>
```

SpringX将bean name和method name聚合在一起，大幅减少了AOP配置代码的行数，其原因在于interceptorNames 属性直接支持多个拦截器。这样配置收拢，更容易理解和维护。它们的AOP配置代码行数对比如图4所示。

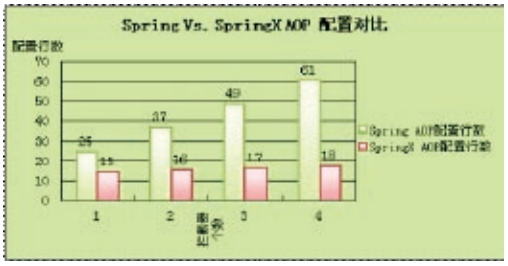


图4　Spring与SpringX的AOP配置对比

结论

SpringX在Spring基本功能上，基于“约定大于配置”的敏捷思想，以及“减少机械配置”、“按需加载”、“快速运行”等敏捷需求，实现了业务bean零配置、敏捷单元测试、敏捷AOP配置三大特性，使大型应用开发更敏捷。

作者简介 | 何坤

阿里巴巴中文站架构师，2009年5月加入阿里巴巴，负责阿里巴巴Apollo项目研发。关注敏捷框架和海量高并发分布式系统设计。目前在研究分布式数据库、分布式ORM、EDA、异步系统等题目。

■ 责任编辑：董世晓（dongsx@csdn.net）

让我们的产品更成功

■ 文 / 马博

作者首先分析了影响产品成败的因素，然后对症下药，从战略、规划、战术三个层面，诠释了做一款成功产品的秘诀。

谈到“成功”和“失败”，我想每个企业管理者和产品的具体参与者都存有“避之不及”与“翘首期望”的双重心态。被冠以“成功”还好，可以上下振奋、内外俱荣。一旦被冠以“失败”，便推搡躲闪、情何以堪。其实定义所谓的“成败”很容易，销量、口碑、利润率、市场占有率、顾客满意度、业界认同、第三方评价等，都可以加冕“成功”的称号，或被冠以“失败”的恶名。但这样简单、宏观或者浅层面的比较我觉得并不客观，也不一定绝对，而且会对企业和产品的具体参与者造成多重的负面影响，对企业和产品的发展都没有太大的益处。所以我个人认为无论是企业、产品的参与者还是其他第三方，都需要理性地对待这个话题。把表象抽丝剥茧、挖掘和分析其核心的因素才是真正值得我们所有人去思考、评价和借鉴的。

影响产品成败的因素

涉及产品的“成功”与“失败”，其根本是产品。产品在每个人的眼中是不一样的东西，比如在企业眼中是传播企业愿景和实现价值交换的**载体**，在产品经理眼中是把企业愿景转化成可市场价值交换的**过程**，在开发人员眼中是技术构想的实现**成果**，在用户或客户眼中是满足需要的**工具**。如果从各自的角度出发看待产品的“成败”，至少会出现很多个不同的组合结果。这样看来，我们就无法从单一角度或者某个独特因素去分析和评价。而其中有很多隐藏的可挖掘的可能性，比如企业战略、产品战略、产品定位、市场策略、人才、渠道、定价、外观、技术等，都有可能是影响个体判断产品成败的因素。每个个体有自己的成败定论，但我认为还是不要简单地看待这个问题，不然失去全面性和客观性的分析和借鉴是基本没有价值的。

我接触过很多企业就是因为没有全面、客观地考虑产品成败的因素，所以败笔产品比比皆是：把可换取利润的产品变成了企业自己的消费品，而且由于思维模式和管理模式的局限性，产品一旦出现问题，很难客观、准确地找到症结所在。这时只能用传统的方式去回溯，简单地用“谁出错谁担责”的方式去加强产品过程中的工作质量。然而这些其实不能根本避免产品出错或者失败的出现，而且还往往因为部门和个人利益等原因，导致互相推卸责任的情况频繁发生。这对企业来说是最不利的，不但从根本上没有解决具体产品上市成功的概率，而且对产品后续的发展也会有深远的不利影响。

我们总会感叹为何国外优秀企业的产品成功率会很高，能达到64%，而国内企业的产品成功率平均只有5%。我不太清楚每个企业是否会统计自己的产品成功率，或者是否对自己产品的成功率满意。但当这个数据摆在我们面前时，我想企业管理者和产品的具体参与者都应该认真反思。因为5%和64%之间有太多的提升空间和利润空间，也有太多的思考空间。

导致产品“成败”的原因还有很多，比如对产品的定义偏差造成的问题。IT企业最常见的一种现象，就是对“产品”本身的定义出现认识错位。比如有些公司把某个功能模块

产品的构成元素

或者应用定义为产品，甚至把某个系统的后台也定义为产品。这样就出现了由于对产品本身定义的狭窄或者偏差导致后续所有围绕产品相关的、系统性的工作和信息没有兼容性和关联性。试想这样的所谓“产品”怎能到市场上完成价值交换，又如何能够保证产品上市后的成功？5%的产品成功率给了我们很肯定的答案。

产品经理：让产品更成功

那么作为企业或者具体产品的参与者该如何思考这些问题、如何提高产品成功率、如何让自己的产品能够成功呢？我就以“产品经理”的视角来做简单阐述。

既然用产品经理的视角来阐述，那就不得不先提提产品经理的根本工作——产品管理。产品管理到底是什么？很多企业和具体管理者都有不同的理解，这是由于企业对产品管理的认识和管理者职业化水平有限造成的。当然也与国内产品管理领域的初级阶段有很大关系。这里我推荐UCPM（中国产品经理联盟）的一篇文章《让产品管理裸奔》，让大家能够更清晰地认识到产品管理的本质。只有理解和认识到产品管理的本质，才能从最科学、客观、准确的角度出发看待产品，才能知道如何把产品做成功。

这里需要明确一个概念。产品是什么？说到底，**产品是承载企业愿景、实现价值交换的一个包含“品牌、服务、介质”的载体，是指具有同一标识的介质、服务、品牌的企业资源集合。**

下面我们从一个真正的、标准的“产品经理”的视角来简单阐述如何做一个成功的产品。

首先要认识到最终体现在市场的产品，在企业内部其实是**“愿景→战略→规划→计划→交换”**这样一个逻辑和流程的最终载体。基于这个思路，站在产品经理的角度需要考虑三个层面的工作：战略层面（产品管理中的战略，非企业战略）、规划层面、战术层面。这三个层面最终可以归结为：知道我们的目标是什么、知道我们该如何去做、知道我们如何能够做到。作为产品经理只要是做产品，就脱离不了这三个层面，而且必须按照这个逻辑和顺序，不然就成了主观臆断或者经验主义。如果顺序乱了，有可能会本末倒置、无序无章。对于企业来讲，如果理解或者理顺了这其中的关系和作用，就可以大大避免出现“失败”的产品。

在这个认识的前提下，我们会发现一个产品**从无到有再到优**这个过程中一定会涉及关于产品的三个层面的具体工作，而且每个层面都是影响下一阶段成败的关键性的活动和前提。

战略层面的工作

对于产品经理来说战略层面的工作尤为重要，这个阶段的工作成果决定了一个产品的目标和方向。试想一个产品如果没有目标和方向结果会怎样。有人会说：“我们的产品在初期也考虑了方向和目标，怎么还是做不好呢？”我来说明一下战略层面的工作内容，各位可以对照一下是否有做过或者是否做到位。

战略层面其实归结为三个方面的工作。

首先是预测问题。预测什么问题？就是我们在做产品之前一定要了解市场、用户和关于产品的一切需求，当然这里的需求是产品管理中的需求，而非研发需求。只有了解了这一切我们才能明确市场是否有需求，规模是否够量级，是否有明确的用户群，用户群在哪里，竞争对手在哪里，是否有相应的资源开拓这个市场等。这就是为什么我们要把预测问题放在首位——它能够为后续的工作提供一个有力的市场依据。

其次是培育机会。我们面对一个市场时要推出一个什么样的产品才能有竞争力，才能可持续发展，才能产生高收益？这就要在培育机会的过程中做一系列的工作，包括评估自身的技术实力、分析竞争格局、明确自己的独特能力、分析管理需求等。只有这些工作都做到位了，才能客观、准确地知道我们想象中的市场

是否真的存在机会，如果存在机会，我们是否真的可能把握住这种机会。

接下来就是产品创新。这里的产品创新并非技术层面的创新，而是基于产品（参照产品的定义）本身和相关资源集合的创新。什么意思？就是我们所关注的创新不仅要在产品“介质”上创新，而且要在影响产品交换过程中的每个环节和元素上创新，包括渠道、包装、价格、概念等。只有这样的创新思想才能保证自己的产品能够更准确、快捷地到达用户，被用户快速接受，这才是创新的目的。当然技术上的创新很重要，但我们要做的毕竟是一个基于商业的“产品”，而不是基于高精尖技术的艺术品，所以作为企业这个商业组织、作为每位创造产品的产品经理来说一定要认识到这一点。

当然有人可能会说iPhone就是技术创新成功的典范。我个人觉得这没什么可比性，从商业环境、企业积累、历史发展都有其独特的因素。而且我们很多人只关注到它技术层面的信息，忽略了渠道、概念、价格、包装、盈利模式等成就iPhone成功的众多因素。如果我们一味强调技术，就有可能走上一条偏离商业规则的非坦之途。当然我不是摒弃技术的纯市场忽悠主义者，我一直认为技术好像人的骨架，其他商业元素好像是人的血肉，骨骼粗壮才能血肉丰满，才能身体健康，缺一不可。

规划层面的工作

规划层面的工作是基于战略层面的工作来开展的，也是把产品战略层面的工作更具体化、形象化的一个关键性步骤。它决定了产品的原始形象和用户群体、决定了我们该如何达到在战略层面中所制订出的方向和目标。

首先是制订产品路线。产品路线的环节中要考虑很多因素，包括价格策略、产品收益和相应的产品所需的一切资源。

其次是产品策略。产品策略工作中需要考虑产品定位、业务流程、阶段性策略等因素。有人可能会问：“这些工作有的做了，有的没做，但都有什么作用呢？”举个例子，比如业务流程方面的工作。一个产品从出厂到用户手里会经过很多的环节，用户首先要知道这个产品，然后又要知道哪儿能买到，知道哪儿卖又要考虑是否选择这款产品等。你可以想象得到一个产品到达用户手中需要经过多少环节，在这些环节中又有多少因素会影响用户对产品的理解和购买。如果不构建一个成本最低、效率最高、最现实合理的业务流程，那么我们的产品再好，用户也很难或者说会花费很高的成本去接受。所以构建业务流程的工作非常重要。当然其他工作皆如此，都会对产品的后续发展产生影响。因此我们必须认识到并重视起来，只有做好每一步的工作才能保证产品的成功。

最后是产品计划。产品计划包括营销计划、控制管理、产品路线等一系列工作，重要性同等重要，这里就不详述。

战术层面的工作

战术层面的工作是现在很多企业和产品经理最常见的。很多产品经理把大部分工作精力放在战术层面的工作中，但不知道在这个过程中都要做哪些具体工作，做到什么程度。他们和研发、UE/UI纠结在一起，整天折腾技术、用户体验、界面美观等这些属于人家专业人士的工作，搞得自己挺忙，其他部门抱怨也挺多。打杂、救火、写文档，也就成为了工作常态。下面我说明一下产品经理如何在战术层面工作、做哪些工作。

首先是概念的实现。这项工作就是要把我们的产品用标准的指标定义出来，简单来说就是定义我们的产品应该是什么样子。

其次是图纸实现。这项工作就是完成产品的相应的规格定义，该阶段最容易出现与技术、UED部门纠缠的情况。这里强调一下，产品经理的规格定义不是具体到该用什么技术实现、用什么颜色表现这些具体的工作。很多产品经理容易混淆产品管理中的规格定义和技术等部门的技术规范，导致工作上的冲突。

接下来是技术实现。这个阶段就是按照之前的规格定义，通过其他部门把产品“介质”制造出来的过程。

最后就是商品和市场实现的工作。我们要在产品上市之前完成一切市场所需要的工具和工作。比如销售工具包、培训计划、产品发布、推广策略等工作。

以上就是对产品经理在做一个产品时需要考虑的几个层面的工作内容的阐述，重点强调了战略层面和规划层面的工作。战术层面的工

作是企业和产品经理经常接触的，但即便很多企业在做也做得不是很到位。究其原因在于我们在推出一个产品之前没有充分考虑前两个层面的众多因素，更没有开展相应的工作，导致后期工作的混乱和效率的低下。这是我们的产品成功率很低的原因所在，也是在分析和考虑产品“成败”所需要考虑的因素。

有的企业可能会说，我们没有考虑这么多也很成功。是的，我不否定，但很多人对“成功”的理解不同。有的企业把某个产品顺利上线或上市就宣布为成功，这我能理解。企业通过投入大量资源（技术、资金等）让产品顺利出现在市场上和用户眼前的那一刻确实值得庆祝，这时大喊“成功”可能有传播的因素在里面。但我们扯着嗓子喊完“成功”以后，是怎样思考如何持续“成功”的呢?

企业或者产品经理推出一个产品的目的，说到底是完成价值交换，让产品利润最大化，而且要持续地利润最大化，最终实现企业愿景。然而据我所知，很多人好像并不明白这一点，依然在模仿、抄袭、无序上马产品。就拿我钟爱的游戏来说，有多少游戏产品是匆匆上马又草草收场？这对企业来说其实是一种极大的资源浪费。当然很多大鳄、垄断型的企业也许不在乎这区区的浪费，不过我还是认为，企业应该在规模发展的同时提高自身的管理水平和认识能力。只有这样，才能保证承载企业灵魂的产品在做到市场表现成功的同时，也在用户心理得到情感认同，这才是真正的成功。

作者简介 | 马博

UCPM（中国产品经理联盟）发起人之一。现主要从事产品管理的研究与实践。曾经涉足重工、广告、会展、互联网、教育等多行业领域。个人邮箱：mabo@chinapm.com.cn。

■ 责任编辑：董世晓（dongsx@csdn.net）

产品管理与产品营销的区别与合作

■ 文 / Marty Cagan 译 / 刘雁 潘希颖 黄捷文

Marty Cagan是享有世界声誉的产品管理专家，曾经担任网景副总裁、eBay产品管理及设计高级副总裁。本文是他回顾自己二十多年来从事软件产品管理工作的总结和经验分享，谈到了产品管理与产品营销的区别与合作关系，最后总结了导致产品失败的常见原因。

产品管理与产品营销的区别

业界权威指出市场上多达九成的产品未能实现既定目标，因而是失败的。即使你的产品不在此列，我依然觉得大多数产品构思拙劣、尚不成熟，可用性差、毫无价值的产品随处可见。

导致产品失败的因素很多，我会尝试从不同角度分析其原因。但我一直认为，最根本的原因是公司对产品经理的职责界定不清，担任这项工作的人缺乏专业训练。我一直在思考这个问题，因为它触及了产品经理的核心工作职责。

产品经理的工作是从细节上定义开发团队开发什么产品。市场营销的职责是对外宣传产品。两项工作天差地别。

为理清职责，我坚持为每款产品指派一名专职的产品经理，负责定义产品（将产品需求和用户体验设计相结合）。然而我发现企业常常会陷入以下三种误区。

由市场营销人员定义产品：由产品营销经理或所谓的“产品经理”负责收集高层产品需求，然后直接交给开发团队开发。这种方式忽略收集详细产品需求的步骤，回避探索（定义）产品的艰难决策过程，也绕开了用户体验设计。

两人分担定义产品的工作：定义产品的工作分给两人完成，产品营销人员负责高层商业需求，“产品经理”负责低层产品需求。

一人兼任两项工作：产品营销人员兼任产品经理的工作（有些公司称这类人为产品经理，有些公司还是叫营销人员）。

下面分别讨论这三种情况及其引发的问题。

由市场营销人员定义产品

这种情况很容易辨认。这类“产品经理”可以为产品团队提供市场营销资源、制作数据表格、培训销售队伍、为产品命名和定价，但是一旦涉及定义产品的具体工作，他们就无能为力，只能袖手旁观。我推荐大家工作之余看看呆伯特（Dilbert）系列漫画，作者用了大量笔墨描绘这类“产品经理”。

这类“产品经理”或许擅长市场营销，但是对详细定义有价值的、可用的、可行的产品往往束手无策。除非他们不但具备营销技能，还掌握管理产品的方法，那么产品还有成功的机会，否则只能寄希望于其他人（比如主程序员、交互设计师、公司高管）挺身而出，担起真正意义上的产品管理工作。然而，更常见的情况是产品从一开始就因此陷入了麻烦。

我第一次接触产品管理工作时，遇到的就是这种棘手的情况，从而导致我以前对产品经理没什么好感。幸亏后来遇到一位贵人，他让我明白了产品经理的真正职责。从那时起，我就开始强调产品经理的作用，并致力于重新定义产品经理的工作职责。

两人分担定义产品的工作

没人单独负责管理产品，这种情况也很常见。产品营销人员（有时被称为“业务责任人”或“商务产品经理”）负责收集高层业务需求；产品经理（在敏捷开发团队中也被称为“技术产品经理”或“产品责任人”）负责收

集低层产品需求。

问题在于两个人都不是真正的产品责任人，没人对最终的产品负责。而且这种模式是基于错误的观点，即认为可以脱离具体需求（尤其是脱离用户体验）定义高层需求。

这种模式让产品经理的工作蜕变成制作各类文档，不但令人沮丧，而且限制创新思维，很难做出成功的产品。

大公司由于业务部门较多，很容易陷入这种管理产品的模式，它们常常为此苦恼，却找不到原因。

一人兼任两项工作

很难找到同时具备产品管理能力与产品营销能力的人。管理产品与推广产品都对产品的成功至关重要，都需要专业的技能，但两者的要求大相径庭。虽然我认识一些能够从容驾驭两项工作的天才，但这样的人少之又少，而且这种团队模式的扩展性很差。即便是最简单的产品，也应该由专职产品经理投入全身心进行管理。让产品营销人员兼任产品管理的工作，即便他具备两种技能，也没有精力把两边的工作都打理好。

开发企业级应用软件的公司，由于非常倚重销售，最容易出现这种问题。销售代表原封不动地把大客户的需求传达给产品经理，再到开发人员。不用说，这样做很难开发出有价值的、可用的产品。

上述三种模式背后都有其原因，认识这一点很重要。很多公司没有意识到错误的模式给它们带来了多大的损失。它们浪费时间，开发出的产品却不是客户想要的，或者只能勉强使用。

解决方法

要解决这些问题，必须清晰界定产品经理和产品营销人员的职责。产品经理负责详细定义待开发的产品，让真实的用户验证产品。产品营销人员负责向外界宣传和推广产品，包括产品定位、产品动态、产品价格，负责产品发布，为拓展市场销售渠道、组织重点营销活动（如在线营销）、促进产品销售提供支持。

请注意，我这里强调产品管理重要性，并不代表产品营销不重要。恰恰相反，我认为产品营销很重要，好的产品营销可以创造巨大的价值。只是这与讨论产品经理职责关系不大。

产品经理和产品营销人员应该经常沟通、展开合作。一方面，营销人员是产品经理获取产品需求的重要来源；另一方面，产品经理是营销人员获取市场营销信息的重要来源。

最后，无论头衔或者组织形态怎么变化，我相信所有成功产品的背后都有一个全权负责定义产品的人。

请记住：如果产品经理定义的产品没有价值，不具备可用性和可行性，无论开发团队多么出色也无济于事。

产品与营销的合作关系

许多公司的产品和营销之间存在问题。这些问题有大有小，严重时甚至会影响公司的正常运作。

从理论上讲，这些问题本不该存在。产品团队努力研发用户喜欢的产品，营销团队试图找到用户，说服他们使用产品。这听起来相当简单，但实际上并非如此。

虽然我强调产品管理和产品营销差异很大，最好由不同的人来担任，但并不是说营销不重要。此外，仅仅物色到产品人才和营销人才还不够，他们还必须有效合作，拿出行之有效的产品方案和营销方案应对市场竞争。

产品和营销之间的合作关系主要涉及以下三点。

首先，为了吸引潜在用户访问网站，必须慎重考虑营销文案的内容，务必做到简单明了、容易理解，同时引人入胜。

其次，充分了解目标用户，高效地把网站（产品）的营销信息传递给他们，最好让用户协助进一步传播营销信息。

最后，在网站上安排具有吸引力的内容，吸引首次访问网站的用户。网站或产品必须符合营销文案的描述，至少要达到基本的用户预期。如果能超出用户的期望，当然更好。

为了获得预期的效果，以上三点都必须处理得当。

如果营销文案描述的内容与实际产品不符，用户首次访问网站后，就不会再回来。这种情况太普遍，可能是因为产品不具备吸引用户的价值主张；也可能是因为营销团队理解的

Marty Cagan

过去20年，Marty Cagan作为负责定义和开发产品的高级经理人为多家一流企业工作过，包括惠普、网景通信、美国在线、eBay。他亲历了个人电脑、互联网、电子商务的起落沉浮，致力于通过写作、演讲、培训帮助客户打造富有创意的产品。为此，他撰写了《Inspired: How to Create Products Customers Love》，创办了硅谷产品集团公司（SVPG）。此前，他的最后一份工作是担任eBay产品管理及设计高级副总裁，负责规划全球电子商务网站的产品和服务。

价值主张与产品经理的有出入（如果是这样，不能全怪营销团队，产品经理可能需要修正产品的价值主张）；还有一种可能是考核营销团队的标准有问题——强调尽可能多地吸引潜在用户（而不是培养忠实的客户），因此营销团队过分看重营销文案的轰动效应。

无论产品多好，如果营销文案不够出色，用户无法充分了解产品，就不会前往网站一探究竟。再好的产品，要写出简单明了、容易理解、引人入胜的文案也非易事。文案既要向潜在用户介绍产品、宣传产品价值，又要避免信息过多，让人无所适从。我建议营销文案只突出1~2个最能吸引用户的功能和特性，不必面面俱到。

即使营销文案和产品都很优秀，如果没人阅读、没人使用，也无济于事。营销团队唯有理解不同目标用户的需求，知道如何将令人信服的产品信息传递给用户，才能制定有效的营销计划，发挥作用。

所以这三点都必须处理得当。你需要合适的价值主张，以及与之匹配的产品，还需要针对目标受众进行宣传。要做好这三件事，产品团队和营销团队必须通力合作。

产品失败的十大原因

我这里所说的“失败的产品”（Weak Product）指的是那些没有实现自身目标，不能为公司创造利润或促进业务发展的产品。

缺乏产品愿景。产品失败的最大原因很可能是产品团队只关注产品功能，没有树立明确的产品愿景，缺少战略性思考。

由其他部门主导的产品路线。产品失败的另一个常见原因是由公司其他部门（如营销部门、客服部门、运营部门、财务部门等）决定采取什么样的产品路线。产品团队往往受到限制，无法完成自己份内的工作。

缺乏快速“尝试—学习”的产品文化。成功的产品来自持续的学习和积累，但很多公司缺少快速“尝试—学习”的技术和文化，而这些对迅速拿出有效的解决方案非常有必要。

糟糕的用户体验设计。好产品靠设计，需要公司有很强的用户体验设计能力。从项目启动的第一天开始，就要考虑交互设计及视觉设计，让设计师在团队中发挥关键作用，而并不是把他们当作团队的附庸。

缺乏对客户的深入了解。很多公司与客户的交流太少，对客户的理解非常浅薄。除了用户调研、用户服务报告、用户讨论会，还应该更频繁地、面对面地与用户交流。产品团队必须成为对用户了如指掌的专家，否则产品成功的可能性不大。

缺少探索（定义）产品的流程。失败的产品公司，一般是由管理层、股东、客户、产品经理提出某个想法（通常是某个功能），然后产品团队把这个“需求”转化成某种形式，交给开发团队开发，完全略过了探索（定义）产品的过程。

缺乏真正的合作。成功的产品要靠用户体验设计团队和开发团队的良好合作。尽早摒弃以下这种偏见：产品经理提出需求，然后设计师设计，最后开发人员实现。在这种情况下，开发人员参与项目太晚，无法提供更好的解决方案。开发人员最清楚什么是可行的，让他们最后介入项目绝不明智。

只关心于发布日期和功能，不关心业绩。公司容易过分关注发布时间，而不是业绩。有可能每周都发布新版本，但业绩完全没有改善。公司关注的应该是业绩。我并不是说要放慢迭代速度，而是强调业绩才是最重要的。

缺乏魄力。成就伟大事业必须冒风险。虽然有方法降低风险（保护企业的品牌和收入），但风险仍然存在。有时是团队不敢于冒风险，有时是领导者。很多企业被恐惧阻挡了发展。

不称职的产品经理。如果产品经理主动性差，执行力弱，对技术和业务缺乏理解，自然会导致产品失败。

当然，实际情况不会是非白即黑，往往有程度上的差别。不妨根据以上十点重新评估你的公司，尝试改变不足之处，增强产品的优势。

本文节选自华中科技大学出版社《启示录：打造用户喜爱的产品》一书和作者的博客。该书从人员、流程、产品三个角度介绍了现代软件（互联网）产品管理的实践经验和理念。特此感谢华中科技大学出版社与Marty Cagan先生授权。

主持人

冯大辉，现任丁香园（http://www.dxy.cn）网站CTO。曾历任支付宝架构师、数据库团队负责人等职。

架构师接龙

岑文初 VS. 杨海朝

提问嘉宾

岑文初

岑文初，2006年加入阿里巴巴，2007年初开始负责阿里软件平台架构设计，2007年底开始进入开放平台领域，2009年8月加入淘宝开放平台，现在是淘宝开放平台主架构师。自认没有特长，只是学习能力比较强。

回答嘉宾

杨海朝

杨海朝，新浪首席DBA，负责整个公司的数据库管理工作。热衷于数据库设计、性能优化、分布式部署方案和高可用性方面的研究。在大规模高并发、海量访问特别是大规模数据库运维方面有丰富的管理和维护经验。

岑文初： 模块化来降低耦合性时如何把握模块划分的粒度？如何权衡复用性与粒度过细导致依赖复杂的矛盾？

杨海朝： 耦合性是影响软件系统复杂程度和设计质量的重要因素，模块化设计的目标是建立模块间耦合度尽可能松散的系统，通过尽量使用数据耦合，少用控制耦合，限制公共耦合的范围和一定要避免使用内容耦合来降低接口的复杂性。

在系统架构中模块化设计对于降低耦合性有非常重要的作用，相关的功能合在一起，不相关的功能分离开来。

模块划分的粒度决定着最终划分结果的合理性和有效性。模块划分的粒度越细，越易于定义和开发，但随着模块数的增加，模块之间的接口也随之增加，使得各个模块在组装成系统消耗的时间增加，同时加大了整个系统测试的复杂度。模块划分的粒度越粗，越有利于整个系统的组装，整个系统的测试复杂度也降低，但整个系统的灵活性下降，难以满足用户对系统的多样性需求。

模块划分的粒度是一个不太容易把握的问题，不同的产品和项目有不同的划分粒度方法。模块化的粒度需要根据模块的三大特征（相对独立性、互换性、通用性）来把握。每个模块的聚合度越高，耦合性就越低，反之亦然。在做模块划分时，应使模块之间尽可能独立，块内联系尽可能大，块间联系尽可能小，功能模块逐层分解和细化，直至形成若干容易编写的模块。同时还需要结合模块的功能、模块的规模、模块的出入口等诸多因素综合考虑。模块划分的粒度越小，通用范围广，可在较多应用进行复用，但其较小的粒度组装过程烦琐，会导致复用效率降低。反之，模块划分的粒度越大，组装的步骤越少，可复用效率越高，但其复用范围受到限制。

岑文初： 如何处理高并发系统中缓存失效的场景？

杨海朝： 从硬件到操作系统，从系统软件到应用软件，我们随处都可以看到缓存的身影。缓存在Web2.0网站中发挥着越来越重要的作用，特别是大流量的高并发系统。缓存在带来性能提升和支持高并发的同时，也带来另一个问题，如果缓存宕机导致所有的缓存失效，所有的流量都压到后端的服务上，例如数据库层。这时，后端服务因为压力过大无法提供服务或快速响应，缓存因为等待后端请求的响应而“热”不起来，最终导致“雪崩”问题。在高并发系统中，解决缓存失效有以下三个思路。

使用Consistent Hashing算法。把数据缓存分片，减少缓存宕机对服务的影响范围。Consistent Hashing能最大限度地抑制Hash键的重新分布，同时要取得比较好的负载均衡的效果，需要在服务器数量比较少时，增加虚拟节点来保证服务器能均匀地分布在圆环上，最大限度地减小服务器增减节点时产生的缓存重新分布问题。

缓存多份。通过应用写多份缓存，或应用写一份缓存由中间件把此份缓存复制多份，在缓存宕机时能切换到另一份缓存上，减少缓存宕机对后端数据存储的影响。这种方式最好采用缓存之间的复制，减少应用开发的复杂度。

解决高并发系统中的缓存失效问题，需要结合业务逻辑进行综合考虑。

实现缓存持久化或半持久化。所谓的持久化就是定期把缓存里面的数据刷到磁盘，保存起来，在缓存失效时能保证大部分数据仍然有效，例如使用Redis或MemcacheDB把一些数据存到磁盘。可能有人会说：“是否在刷磁盘那一刻会影响缓存的高效性？”这可以考虑通过高端硬件（例如Fushion IO或SSD作为存储设备），来减少刷Cache过程中减少对服务的影响。

这些仅仅是实现的三个思路，各有利弊。解决高并发系统中的缓存失效问题，需要结合业务逻辑进行综合考虑。

岑文初： 目前是如何做好应用的依赖监控的？例如依赖缓存，是否知道缓存的命中率情况、使用率等；依赖外部服务，是否知道外部服务的消耗时间、成功失败情况等。

杨海朝： 应用监控日益成为保障系统安全运行不可或缺的工具，在提高系统可用率和故障的预警能力、最大限度地缩短故障修复时间方面，日益显示出其重要程度。

此外，对应用依赖的服务的监控也非常重要，目前可以建立一整套监控体系，这套监控体系不仅监控应用自身，同时也对外部服务进行监控。应用在调用依赖服务时，打印应用调用的详细日志，记录调用的返回时间，以及成功与否，同时要求外部或依赖服务提供一个统一的接口，用来返回每次调用消耗的时间以及失败率。通过对这些日志的实时分析来发送邮件和短信报警。

岑文初： 系统是否有对内或者对外提供服务，如何管理服务的使用权限，如何做服务升级？

杨海朝： 在系统中服务的使用权限至关重要，特别是既提供对内服务又提供对外服务的系统。

目前管理服务的使用权限，设置白名单是最简单的方式。通过设置白名单，只有在白名单里面的用户和来源IP才能访问服务，其他的都无权访问。外部调用服务也通过使用appkey的方式，简单地说是API接口的钥匙，只有通过这个钥匙才能打开API的大门，从而获取数据，同时对appkey的申请设置一系列流程来保证其安全性。根据这些appkey来设置不同的权限等级，控制对目标数据的访问以及调用频率。

通过使用Gearman系统进行“热”升级。在负载均衡设备的后端增加一个代理层，升级时通过对代理层的心跳探测页面修改，把落到这台机器上的流量切走，然后对这台机器进行升级，升级完之后进行测试，测试完成之后，再对某一区间机器进行升级，在某一个时间内测试没有问题，再对其他所有区间进行升级。提前准备好回退机制，保证每次升级对用户的影响最小，力求达到无缝升级。

岑文初： 对于不同系统之间的耦合如何处理？比如前端系统要求快速响应大量的用户请求，但是依赖于后端的服务体系，而服务系

统的瓶颈可能在数据库读写上，如何协同这样需求差异化系统工作?

杨海朝： 不同系统之间的耦合性是指各个系统之间的互相依赖程度。需要熟悉各种耦合性的特性，为不同的系统之间选择合适的耦合性。

如果系统之间采用紧耦合，那么涉及对象与对象的直接通信，这些对象通常比在松耦合系统中交互更为频繁。假如两个对象位于不同系统中并且由不同的网络分开，则会导致性能和延迟问题。

如果系统之间采用松耦合以及基于消息的架构，客户端和服务端不需要知道对方如何实现。两端的消息协议符合协商，则客户端或服务器端的实现就可以根据需求进行变更，而不会互相影响。

松耦合提供了紧耦合所不能提供的许多优点，并且它有利于降低客户端和服务端之间的依赖性。例如采用异步策略。对于需要快速响应的系统，尽可能把过程变成异步的，减少请求者所经历的响应延时。如果系统A同步调用系统B，那么A和B是紧密耦合，而紧耦合的系统是需要共同进退，两个系统需要承担相同的量，并且如果B不可用，则A也不可用。如果系统A和系统B之间是异步的方式，不管是通过消息队列、多播消息、批处理还是其他实现方式，那么系统A和系统B是相互独立的，如果B宕机，A仍然能够继续提供服务，也即所谓的优雅降级策略。这种异步策略能把高并发的写入高峰变成一个平缓的写入速率，减少了硬件的部署成本，同时可以控制消息队列的写入速度以及优化队列来避免后端服务出现压力过高问题。

岑文初： 对于系统优化，平时如何查找瓶颈，最后又如何确定优化后是有效的？可以的话请举例说明。

杨海朝： 系统上线，首先对系统进行必要的压力测试，量化能承担的最大量。通过对客户端性能指标和非客户端性能指标进行收集，对系统的响应时间、并发用户数、吞吐量、硬件资源的表现等这些关键点进行分析，最终找到系统的性能瓶颈。

同时通过这些压力测试，确定系统在访问量不端增加时，可能出现瓶颈的组件。压力测试过程中输出详细的日志信息，然后对日志信息进行整理，分析各种组件的性能表现情况。

对于已经在线的系统，要部署好各种状态监控系统，最好以图形化的方式呈现出来。在出现性能瓶颈时，能通过这些状态监控信息，找到性能瓶颈点或将性能瓶颈缩短在一定的范围之内，然后对线上系统按逻辑功能进行划分，例如静态池、动态池、缓存层、数据库服务。修改每个功能模块以便输出详细的日志信息，对这些日志信息进行整理和分析，找到瓶颈所在，同时记录各个功能块的性能表现，在优化后，收集相同信息进行对比来确定优化前后的性能差别。对于单个模块可以通过一些性能分析工具来对程序的性能进行分析，例如Rational Quantify。

下面以数据库的瓶颈分析为例。

- 收集一天内所有的SQL语句，以及统计每种类型的语句占的比例。
- 收集当前系统中CPU、内存、硬盘I/O的表现情况，分析硬件资源瓶颈。
- 收集满日志并对其进行分析整理。
- 确定每条SQL语句在数据库中的平均执行时间。
- 对执行效率低、消耗硬件资源、响应慢并且频繁调用的SQL语句进行优化。
- 将收集优化后的硬件资源表现情况和之前进行对比，最终量化优化提高的比例。

总之对于不同的系统，寻找性能瓶颈的步骤可能会有所不同，但都离不开对关键数据的收集和分析。

■ 责任编辑：董世晓（dongsx@csdn.net）

一分钟先生
Mr. One Minute

如何做好企业/团队的技术选型？

好的技术选型，能最大程度地提高企业和团队的效率，从而开发出满足用户需求的产品。作为一线的技术管理者，他们都是怎样做的呢？

冯大辉

丁香园网站CTO

大公司或者大一点的团队的技术选型几乎不需要太多讨论，因为最后会不可避免地绕到技术官僚的话题上去。这里我简单说说技术型创业团队的技术选型问题。

拥抱开源技术

如果只能选择微软的技术路线，比如团队只会用微软技术，也不想学别的，那么似乎没有别的办法，将就一下吧。如果还有的选择，尽量使用开源技术。这样不但可以有效降低软硬件成本，还有更多的部署方案可供选择，服务器上线甚至还能避免病毒的侵袭。开源技术的好处是出了问题，你总有办法可以找到答案。而用微软的产品，可能平时不出问题，但一出问题，你根本没什么办法。微软的产品使用门槛倒是低，但复杂度可一点都不小，而且随着发展，成本越来越高。国内有几个大中型网站，比如天涯、5173、大众点评、京东等，怕是深有感触吧，有的因为成本太高而继续被捆绑，有的则破釜沉舟要摆脱这种束缚，但不管怎样，总要付出一定的开销。

开源技术路线有数种分支，该怎样选择呢？选择大路货，选择可以掌控的开源技术产品、语言、程序、框架，乃至解决方案。比如PHP，比不上Ruby阳春白雪，但用户基数大，总能找到不错的工程师。PHP虽然粗糙，但是管用。以PHP作为开发语言的成功产品不计其数，很多东西根本不需要你再开发，稍加定制即可。技术本身没有高下之分，差别在于使用技术的人。

避免过度炫技

技术人员创业最容易犯的一个错误就是“炫技”，即热衷于使用最新、最时髦的技术。新东西的确可以给技术人员以满足感，但也很快会将你的时间资源消耗进去，除非你准备做的是一款基础产品，否则你要花时间去学新规范、熟悉新功能、对付新出现的软件Bug……但这时你最需要做的是开发产品，而不是捣鼓其他东西。一些新技术或者方案，可以花些时间分析一下但没必要立刻就用，只需确保将来有一天能真的用上时，对一些重大的陷阱或是缺陷能够了然即可。

很多人神往37Signals的成功，但你一定要知道类似37Signals的团队，默默无闻地夭折掉的不知道有多少。每当看到创业团队就那么1~2个人还整天捣鼓Go、Erlang这些东西，并想硬生生地用到产品中去，我就知道，这样的团队要悬了。有这些精力和能力，应该想办法尽快让技术变现，研究一下怎样改进产品、怎样给用户带来更大的价值，而这些不一定用最好的技术才能做好。想办法尽快让产品发布，尽快接受更多人给你第一轮反馈，只凭创业团队几个人闭门冥想是很难出来好产品的，有时产品推出的时机比完备的功能更重要：GroupOn最早不过是搭建在WordPress上的几个页面，而阿里巴巴网站最初也不过是一个论坛，你又何必等到所有细节都打磨好呢？

拥抱开源技术，避免过度炫技，如果技术型团队创业（做互联网），这两条都能坚持的话，我想你已经抓住了问题的80%的部分，基本上你不会做太多的无用功。

另外，刚启动时不要直接招技术总监、技术经理、架构师这些看起来级别很高的人，因为他们未必认同你的想法和你现在的团队。建议找能实现你产品想法的人。最后有一点必须要说一下：不要因为一个人的技术喜好而舍弃整个技术团队，在任何时候这都是很愚蠢的事情。

冯兴智

普元信息资深架构师

在重大产品决策或者大规模应用开发前一般需要进行技术选型，其目的是为了降低产品研发的技术风险。所以首先需要明确为什么需要技术选型、需要达到什么目的，整个过程需要有一套的组织流程来保证。

一般可以将整个过程分为调研、候选对比、关键技术验证、原型验证几个阶段。在调研阶段主要调研对象是目前该范围业内主要产品以及开源产品，需要了解其主要技术特点和各自的优势和劣势；在候选对比阶段，是在前一阶段基础上选出两种倾向用于最终路线的技术进行进一步的研究和对比。在关键技术验证阶段，需要列出所有的技术验证点，对验证点描述、验证方法、验证步骤、验证前提、验证环境以及最后的交付物和评估方法指标在验证方案中体现；在原型验证阶段，主要是针对重点关注的场景，通过一个原型来整体验证比较。在这个阶段一般需要进行概念模型、编程模型以及结构设计例如设计时视图、系统结构图等，定义需要的API，必要时还需要划分子场景，在场景中包括场景描述、Feature、预研点以及相关设计。

当然也需要对人员角色进行分工，一般划分程序经理、开发人员、测试人员几种角色。程序经理需要确定原型目标范围，编写原型目标文档并组织评审；制订和跟踪原型开发计划，对原型进行验证，确保原型符合原型目标要求。开发人员需要从开发角度提出原型需求，评审原型目标文档，按照原型目标文档和原型开发计划完成原型相关设计和开发。测试人员需要Review原型目标文档，根据原型目标文档采用一定的测试手段验证原型是否符合原型目标要求。

在最近我们进行的ESB新产品中，就采用了类似上述的流程。我们确信技术选型的最主要目标是要自主掌控，所以从非常底层的技术开始，重点关注并发、资源隔离这样的目标，解决在不确定环境中实现交易控制和可扩展的目标。所以我们设计了一个三段式的SEDA架构，基本原则是要实现架构的资源可分配性，提升吞吐率，当然最初实现的功能可以比较简单。通过上述流程，有效保证了我们在新产品研发过程中的效率和产品架构质量。

最后，在技术选型产出物上，一般的结果可以有三类，有马上转化应用的新产品，也可以是结合现有产品的一个解决方案，或者是某个方面的一个提升点（如一个新的组件）。

石钰

奈特软件联合创始人、首席架构师

关于公司/团队技术选型的话题涉及到很多非技术层面的问题，特别是大公司在技术选型方面，不可避免要受到企业官僚的影响。

下面我结合自己公司的实际情况，谈谈初创企业或小型开发团队的技术选型。技术选型涉及产品/项目开发流程中各个环节所用到的工具和技术。例如项目开发前期的需求收集、整理分析工具、开发阶段的IDE和版本控制系统、测试阶段的测试工具等。

我们作为初创企业在进行技术选型时会根据产品的需求、技术的复杂性、可扩展性、跨平台可移植性以及成本来做出最终的决策。

成本因素

初创企业的资源特别是资金往往不是很充裕，如果采用商业化的技术解决方案，往往价格不菲，像Visual Studio专业版、Windows Server、SQL Server的价格动辄上万，这种情况下不妨考虑一些免费开源的技术框架。比如使用SharpDevelop代替Visual Studio，MySQL代替SQL Server。此外，也可关注并加入一些针对初创企业的扶植计划。例如我们创业前所工作的公司采用的都是微软技术，大家对微软技术掌握得很熟练，这样自己成立公司开发自己的产品时同

样优先考虑微软技术。我们通过加入微软的BizSpark计划，有效降低了成本。

复杂性，成熟度

我们在做技术选型时最忌讳的就是盲目追崇新技术和框架。有些技术刚刚面世不久，还没有成熟的社区支持和成功案例。如果这时为了赶时髦或者在产品的推广宣传上加点花头而采用新技术的话，往往会导致项目陷入泥潭。采用不成熟技术而导致失败的案例比比皆是，如网景在Java刚面世不久就使用Java重写Netscape Navigator浏览器，导致用户体验大幅下降及用户群的流失。还有当时被称为下一代Windows客户端技术的WPF，到现在发展得仍然不瘟不火，如果仅仅是为了更炫的展现效果而使用WPF，结果肯定会得不偿失。

产品自身因素

技术选型不可避免地要以产品为中心。如果产品是Windows智能客户端软件，那么微软的技术框架必然成为优先考虑。如果开发Android软件，Eclipse和Android SDK同样必不可少。

刘舒

阿里巴巴高级产品经理，曾任微软测试开发工程师、腾讯测试经理

我在微软和腾讯从事了三年的软件测试和团队管理工作，期间涉及两大类的研发工作：第一，用于质量控制的各种平台系统，例如缺陷管理系统、用例管理系统、产品健康指数可视化系统、自动测试管理系统等；第二，用于具体的产品测试工具，例如桌面软件的UI自动化测试工具、JavaScript自动化测试工具、Web服务压力/性能测试工具等。

在团队的建设和技术选型上，遇到过一些困惑也走过一些弯路，总结出来有两点经验。

阅读公司文化，借鉴成功团队经验

测试团队一定要深刻阅读和理解公司文化，作为其技术选型的首要考虑因素。比如，在微软，无论是构建质量控制系统，还是开发各种产品测试工具，都以Windows+IIS+.NET作为技术框架；在腾讯，就会选择LAMP这样的开源技术框架。

在选定了技术框架的基础上，还面临一个问题，就是应该如何去设计和实现出具体的系统或者工具。一个非常重要的经验就是一定要借鉴成功团队的经验，站在巨人的肩上。例如，当初在腾讯，我们需要评测QQ地图的POI检索功能的质量和用户满意度。于是，我们花两周时间设计了一个自以为完美的盲测系统，结果在评审时才发现该盲测系统功能过于简单而且性能达不到指标。后来，我们在与腾讯SOSO团队讨论时，才发现他们经过多年的研发，已经有一款非常完善的搜索盲测系统，直接借鉴过来，就能很好地解决项目测试的问题。这点特别是对很多新任命的团队负责人，是很重要的经验。

敏捷务实，持续集成，切勿过设计

工程的基本要素是实用和强调执行力。对于一个软件/互联网企业或者团队而言，最重要的是先解决眼前遇到的具体问题，而不是盲目追求系统的扩展性和性能。例如，我们在测评QQ地图后台服务的性能状况时，首先想到的是选用的测试方案要具备良好的功能特性，可扩展性要强、足够开放，能够与已有的一个研发管理系统做集成。按照这个标准，我们选择了LoadRunner，这是一个比较复杂的工具，光配置就花掉了大量的时间。结果，导致实际测试的时间太靠后，发现的性能问题都无法在即将发布的版本中得到修正。这就是一个非常典型的因为过设计而引发的技术选型耽误工程进度的例子。后来，我们选用http_load这个开源、简单、实用的工具对后台服务做快速的压力测试，从而在第一时间得到测试报告。然后在没有测试任务的时候，花时间将该工具集成至研发管理系统中，做到了持续集成。

其实总结起来，测试团队的技术选型首先要顺应公司的框架性要求，然后在具体实施过程中，要以解决问题作为决策目标，多向成功团队取经，敏捷务实，切勿过设计。

程序调试与啤酒

——Erlang之父Joe Armstrong访谈

■ 文 / Peter Seibel　译 / 米全喜

以啤酒收取程序调试报酬

Seibel：你是如何开始学习编程的？是从什么时候开始的？

Armstrong：是从中学时开始的。我出生于1950年，上中学那会儿还没有几台计算机。到了中学最后一年，那年我应该是17岁，我们当地的议会得到一台大型计算机，好像是IBM的。我们可以在上面写Fortran程序。通常，我们在编码纸上写好程序，然后发出去。一个星期后，等编码纸和穿孔卡拿回来的时候还必须确认一下。但是制作穿孔卡的人总会出点错，所以可能要反复1~2次才能弄好。最后这些穿孔卡就可以送到计算机中心了。

卡片进入计算机中心后会再拿回来，因为Fortran编译器会在程序中出现第一个句法错误的地方停下来，后面的程序就都不处理了。你的第一个程序似乎需要3个月才能跑通。我认识到，不能每次只送一个程序，应当并行地开发多个单一子例程并且一次都送去。我记得写过一个显示国际象棋棋盘的小程序，用打印机绘制出来。但是因为中间等待的时间太烦人了，我不得不把所有的子例程都当做并行的任务一次写完。

Seibel：你学的是物理学，是从什么时候开始转向编程的？

Armstrong：嗯，有一些本科生的课程需要编写程序，而我又特别喜欢编程。我还非常善于调试程序。如果别人程序出了问题，我就会去调试别人的程序。标准调试的开价是一杯啤酒。也可能提价，还有两杯啤酒、三杯啤酒的问题。

Seibel：在给他们调试程序时，是以他们必须给你买多少杯啤酒而论的，对吗？

Armstrong：对，等我修复了程序时他们要给我买啤酒。我在读程序的时候总是在想：“他们为什么要这样写程序呢，太复杂了。”我会重新编写并简化程序。看到人们编写复杂的代码我感到很吃惊。有些问题用几行代码就能解决，但是他们要写上几十行。我有点好奇，他们为什么看不到简单的方法呢。而我就颇为擅长采取简单的方法。

我真正开始编程是拿到第一个学位并打算读博士学位的时候。我开始读高能物理博士学位并加入了那里的气泡室小组，他们有一台计算机。那是一台DDP-516，是Honeywell公司的。我可以独自一人使用它。它是穿孔卡式的，但是可以在上面直接运行程序，只要把穿孔卡放进去，按一下按钮，答案刷地一下就出来了。我特别喜欢那台计算机。我在上面编写了一个小象棋程序。

那时的实际磁心存储器是由妇女编织而成的，能够看到磁心和一块块的小磁铁和穿进穿出的线路。它的价格高得惊人，有一个大约10MB的磁盘驱动器，上面有20个小底板，大约15公斤重。它还配了一个电传文本的界面，可以在上面输入程序。

后来又出现了“玻璃电传打字终端”，那是最早的视频显示器设备，可以在上面输入并编辑程序。我觉得这太神奇了，再也不需要穿孔卡了。我记得当时和计算机管理员说：“要我说，将来有一天人人都会有这样一套机器。”他说道：“我看你疯了，Joe，你真是疯了！”“为什么不可能呢？”“这些东西贵得离谱。”

正是从那时我真正开始学习编程了。当时我的导师对我说：“你不应该再读物理学博士了，改行吧。你热爱计算机，应当搞计算机。”我说道：“不，不，不。我不能半途而废。”但实际上他的话是对的。

Joe Armstrong
Erlang编程语言发明者，创建了开放电信平台OTP。他起初是一名物理工作者，在攻读物理学博士学位时因积蓄用完而转向计算机科学，找到一份研究员的工作，为英国人工智能领域奠基人之一Donald Michie工作。在Michie的实验室，Armstrong接触了人工智能领域各个方面的杰作，成为英国机器人学会的创始成员并撰写了一些有关机器人视觉的论文。由于Lighthill的那份调查报告，人工智能的资金来源枯竭，Armstrong又转回从事了5年多与物理学编程相关的工作。开始时他在欧洲非相干散射科学协会（EISCAT）工作，后来又到了瑞典空间研究中心，最后加入了爱立信计算机科学实验室，Erlang就是在那里发明的。

不同寻常的工作经历

Seibel：那你拿到博士学位了吗？

Armstrong：没有。我没钱了，所以没有读完。我后来去了爱丁堡大学。此前在读物理的时候我们常常到物理系图书馆去学习。在图书馆的角落里有一些计算机科学书籍。有一些棕色封底的杂志叫做《机器智能》，一共有4期，是爱丁堡大学的机器智能系编辑出版的。我学的是物理学，但我却渴望阅读这些杂志，并且在想："真是太有趣了。"Donald Michie那时担任爱丁堡大学机器智能系的主任，我给他写了封信，说我对这种东西非常感兴趣，问他那里有没有工作可做。他给我回了信，说目前还没有，不过无论如何，他很想和我见一面，看看我是什么样的人。

几个月后我接到一个电话，也可能是一封信，是Michie的。他说："我周二去伦敦，我们见一面如何？我要乘火车回爱丁堡，你能来车站吗？"我去了车站，见到Michie，他说："嗯，不能在这儿面试——我们去找个酒吧。"于是我们到一家酒吧聊了聊。过了没多久又收到他的一封信，说："在爱丁堡大学有一份研究工作，你申请一下吧。"于是我成了Donald Michie的研究助理并去了爱丁堡大学。我就这样从物理学转到了计算机。

Michie在二战期间曾经和图灵在布莱切利公园（Bletchley Park）（编者注：二战爆发前，英国在距离伦敦不远的布莱切利公园设置了国家密码破译机构，许多破译员在那里工作，破解德国电报密码）一起工作过，拿到了图灵所有的论文。我在图灵图书馆有一张书桌，周围也全都是图灵的论文。我在爱丁堡大学待了一年。此后由于数学家James Lighthill的原因，爱丁堡大学都有点维持不下去了。Lighthill受雇于政府，前往爱丁堡大学调查人工智能。他回去后说道："那个地方什么有商业价值的东西也弄不出来。"

说得就像一个巨大的儿童游戏区。我是英国机器人学会的创始成员，我们都认为这个工作意义重大。但是拨款机构却说："机器人！这是什么东西！我们不打算在这上面投入资金了。"我记得那是1972年前后，所有的资金来源都枯竭了，大家都说："嗯，在这里度过的时光非常美好，但现在最好还是找点别的事情做吧。"

于是我又回去从事物理学工作了。我到了瑞典，在EISCAT科学协会得到一份物理学程序员的工作。我的上司来自IBM，年纪比我大，他想要上头给出一份规格说明书，这样他就可以拿去实行。我们曾经讨论过这个问题。他说："如果没有任务说明，也没有规格说明，这样的工作太糟糕了。"我说："嗯，如果没有任务说明，那才是一个好任务。因为你可以按照自己喜欢的方式来完成。"一年后我的上司离职了，我接替了他的工作，担任首席设计师。

我为他们设计了一个系统，可以称为应用操作系统，那是一个在普通操作系统上运行的系统。那个时候计算机的价格已经比较合理了。我们有一些NORD-10计算机，是挪威制造的——我觉得他们这种型号的计算机想要进入PDP-11的市场。

我在那里工作了将近4年。接着在瑞典空间研究中心得到一份工作，构建了另外一个应用操作系统，用于控制瑞典发射的名为"海盗"的第一颗卫星。那是一个有趣的项目，不过我忘了那台计算机的名字了，只记得它克隆的是Amdahl公司的计算机。那上面还只有行编辑器，没有全屏幕编辑器。所有的程序都只能放到一个目录下面。文件名是10个字母，扩展名是3个字母。还有一个Fortran编译器或汇编语言编译器，全部东西就是这些了。

有趣的是，现在回头想想，我不认为当今这些小玩意会让你的生产率更高。比如说分层文件系统，它怎么可能让生产率更高呢？很多程序开发方式是在脑海中形成的。我认为在那些简单的系统上工作可以强制你规范地进行思考。如果没有目录系统，就只能把所有的文件都放到一个目录下面，你只能变得相当规范。如果没有修订控制系统，你也只能变得相当规范。如果自己做的事情能够规范起来，那我觉得分层文件系统和修订控制系统也就没什么可取之处了。它们解决的问题并不能从本质上解决你的问题。如果多人一起工作，它们可能会让事情容易一些。但对个人来说，我看不出来有什么差别。

另外，我觉得我们现在因为选择过多而不堪重负。我的意思是，那时候我只能使用Fortran。甚至连Shell脚本也没有。只有可以运行程序的批处理文件，编译器，还有就是Fortran。如果确实需要的话，还可能有汇编语言编译器。不需要痛苦地做出选择。今天年轻的程序员肯定会感到很

不舒服，面对20种编程语言和几十种框架，该如何选择，真是无所适从。我们那时没有这些难以选择的地方。只要开始做就行了，因为使用什么语言、什么工具都已经是定下来的。不需要考虑该做些什么，只管做就行了。

打开黑盒的重要性

Seibel：另外一个差别是现在也无法彻底了解整个系统了。也就是说不仅仅是要做出很多选择，而且在选择要使用哪些黑盒的时候，还不一定完全理解黑盒的工作方式。

Armstrong：是啊，如果这些大黑盒不能正常工作，必须做出修改，我觉得自己把所有的内容都从头开始编写一次会更容易些。做不到软件复用，真是太糟糕了。

Seibel：但是如果把所有这些黑盒都打开，看看里面有什么，看看它们的工作方式，再确定如何对它们做一点改造来满足自己的需要。你觉得这样做确实是可行的吗？

Armstrong：这些年我犯了一些人们常犯的错误，那就是没有打开黑盒。有时候想一想，觉得这个黑盒无法理解，难度太大，所以不想打开它。我曾经打开过1~2个黑盒。有一次我需要做一个窗口系统，为Erlang做一个图形系统，我在想：“嗯，就在X Windows上运行吧。” X Windows是什么呢？它是一个套接字，上面跑着协议。只要打开套接字，往里面注入这些消息就可以了。为什么要用库呢？Erlang是基于消息的。整体指导思想是向其他东西发出消息，让它们执行操作。嗯，X Windows中的指导思想则是，有一个窗口，向窗口发送消息，再由窗口执行操作。如果在窗口中执行操作，它会把消息回送给你。这非常像Erlang。但是X Windows的编程方式是运用回调库——如果出现了这个情况就调用这个函数。这不是Erlang的思考方式。Erlang的思考方式是，给某个东西发送消息，让它做一些事情。所以，等一下，把其中的库去掉吧，直接和套接字对话。

猜猜结果会怎么样？非常简单。X协议收到了一些消息，我不知道具体是多少条，也许是100条、80条，大致就是这么多。但实际上只需要其中的20条就能完成有用的工作了。把这20条消息映射到Erlang术语上，变个小魔术，然后可以向窗口直接发送消息，它们就开始执行动作了。这样做的效率也很高。但界面不是很好，因为我没有把太多的精力用到图形和艺术标准上。如果为了让界面再美观一些，要做的工作还很多。但是不管怎么说，实际上并不难。

另外一个例子是我做的排版系统，我打开的抽象边界是PostScript。到了边界的地方你会想：“我不想越过这个边界。”因为你会认为边界里面的东西极其复杂。但是我再次发现，它实际上是很简单的。那是一种编程语言，一种不错的编程语言。抽象边界很容易穿越，而一旦穿越，会有很多收益。

这些年来我注意到，真正好的代码是我完全进入状态的时候编写的。如果注意力不集中，就停止编程，做点别的事情。

在出版我那本Erlang编程书时，出版社说：“我们有画图工具。”但是画图工具真的很难精确地对准箭头，所以我不喜欢那些工具。而且画图的时候手也很难受。我想：“编写一个生成PostScript的程序，然后在‘这里画个圆圈、那里画个箭头’，让程序正常运转起来，这样一比，编程花的时间并不长。”编写程序需要几个小时。以所见即所得的方式画图也需要这么长时间。只是自己编写程序还有两个好处。你的手不会难受，而且即使把图形放大一万倍，看到的箭头也是对得整整齐齐的。

我并不是说刚入行的程序员应当把所有这些抽象的东西都打开。我的意思是，一定要考虑是否可以打开它们。不要完全放弃这个想法。看看直接到达的途径是不是比包装后的途径要快一些，这是值得一看的。一般来说，如果购买软件或是使用其他人的软件，一定要充分考虑还需要花很长时间来加工这套软件，因为它和你想要的不完全一样。软件的执行方式有微妙的差别，而这个差别可能需要很长时间来解决。

编程这些年的变化

Seibel：同刚开始编程时相比，你在看待该如何编程的问题上最大的变化是什么？

Armstrong：我认为编程方式中的最大变化与硬件无关。显然，现在的计算机速度要快得多，功能要强大得多，但是人的大脑比最

好的软件工具还要强大一百万倍。我在编写程序的时候，几天之后会突然说：“程序中有一个错误——如果这样、那样、那样、这样的话，程序就要崩溃了。”然后我去看了代码，确实如此。此前一点征兆也没有。你能告诉我哪一个开发系统能够做到这一点吗？作为一个程序员，我所发生的变化是内心思想的变化。

我认为在经过多年的编程之后会有两个变化。一个变化是，在年轻的时候，我会不停地写程序，直到完成。当程序完成后，我就不再管它了。程序写好了，完工了。然后我会突然领悟：“啊！搞错了！真是笨蛋！”我会重新编写程序，后来再次发现：“噢，程序是错的。”于是又重新编写。

我记得当时有这样一个想法：“先不要动手写代码，把这些东西都想好，这样做不是很好吗？”如果我不写代码就能获得那番领悟，不是很好吗？我认为现在可以做到这一点了。那20年可以算作是学习如何编程的时期。现在知道该如何编程了。我以前通过实验来学习编程。现在我知道该如何编程了，不需要再做实验了。

偶尔，我也得做一些很小的实验，比如编写一些非常小的程序来回答某个问题。我会把事情想清楚，等到开始编程的时候，这些程序就可以或多或少地像我预计的那样运行起来了，因为之前我已经想清楚了。这也意味着要花很长时间。编写程序、有所醒悟、重新编写。这样可能需要花上一年的时间来写程序。所以我现在可能不这样做，而是先思考上一年。我不会再做那种简单的输入工作。

这是第一个变化。出现的第二个变化是直觉。在年轻的时候，我会通宵地写程序，干到凌晨4点钟，精疲力尽，那是男子汉气概的编程，一个小时接着一个小时，不停地编写代码。即使情况不好我也坚持不懈，总要让代码能够跑起来。即使没有直觉，我也要继续编程。

我得到的教训是，在疲惫的时候编写的程序都是垃圾，第二天就要把它们都扔掉了。20年前，就算强烈地感到事情不对劲、代码中有错误时，我也会继续编程。这些年来我注意到，真正好的代码是我完全进入状态的时候编写的，时间不知不觉地过了，而我甚至没有在考虑程序，只是很放松地坐在那里，输入这些东西，看着自己输入的东西出现在屏幕上。这样的代码会很不错。如果你不能集中注意力，弄出来的东西会说：“不行，不行，这儿错了，那儿也错了。”可我在多年前并没有注意到这一点。写出来的代码都被扔掉了。现在，如果觉得不行，我就不再编程了。“不能再写了。”这是我根据经验得到的，停下来，不要再写代码了。不要再处理这个问题了。干点别的。

我在上学的时候很擅长数学之类的课程，所以在想：“噢，我是一个按照逻辑思考的人。”但是我参加心理测试时，在直觉上得了高分，而逻辑思考方面的分数却有点低。不是很低，我还是可以做数学这类的题目，我相当擅长。但正是因为我擅长数学，所以我过去认为科学是关于逻辑和数学的。我现在就不会这样说了。我要说科学也有很多直觉，根据直觉能够知道什么是正确的。

Seibel：你现在在编码之前会花更长的时间思考，那么在思考阶段会做些什么呢？

Armstrong：噢，我会记些笔记，我不仅仅是在思考。在纸上随便写点什么。我可能不会写很多代码。如果你密切注意我的活动，会发现我大部分时间都在思考，偶尔写点什么。另外一件对解决问题非常重要的事情是问问我的同事：“你将如何解决这个问题？”你找到他们，说：“我不知道应当采取这种方式还是那种方式。必须在A和B之间做出选择。”然后你向他们描述A和B，等讲到一半的时候，你会说：“啊，是B。谢谢你们。非常感谢。”这样的事情发生过很多次。

你需要这样一块智能白板，如果你只是独自一人在一块白板上写写画画，是得不到反馈的。但是如果面对的是人，你会在白板上向他们解释替代方案，他们也会加入讨论，提出一点建议。然后突然间你就知道答案是什么了。对我来说没有涉及到代码编写。但是和处理同样问题的同事进行交谈是非常有价值的。

本文节选自人民邮电出版社北京图灵文化发展有限公司出版的《编程人生》一书。该书是当今15位大师级计算机程序员的访谈录，重点介绍了他们的编程感悟。特此感谢图灵公司授权。

【编者按】许多大师和业界专家都非常强调技术论文的学习，我们近期采访的图灵奖得主Chuck Thacker甚至将阅读历史性的经典文献作为自己的成功之道。2011年开始，我们特别开辟“论文研读”栏目，欢迎大家推荐有价值的论文。

Ceph：一个可扩展的高性能分布式文件系统

■ 文 / 马如悦

最近百度完成了对Hadoop HDFS元数据管理（NameNode）的分布式改造工作。在改进过程中，借鉴了很多Ceph的设计思想，所参考的论文是发布在OSDI06上的《Ceph：A Scalable, High-Performance Distributed File System》。本期以Ceph的分布式元数据管理实现作为引子，全面介绍了GFS、Lustre、HDFS在这方面的实现。

百度为何要启动Hadoop NameNode的分布式化改造？以百度之前的一个HDFS集群为例，集群规模为700台机器，存储的文件数接近5000万，块数也接近5000万，容量占用85%，使用的内存从监控页面显示已经占用将近30GB的物理内存（包括JVM占用的空闲内存但没有释放给操作系统的那部分）。从发展趋势来看，当集群继续扩大到5000台左右时，文件数接近3.5亿，块数接近3.5亿，内存占用接近210GB。当时CPU负载从NameNode的Ganlia监控来看，8核的机器CPU占用为10%，如果扩大到5000台集群的规模，可能造成8核都跑得很高（由于锁的存在，NameNode也不能完全将8核利用起来，所以5000台时可能有些请求就已经不能及时处理）。从内存压力和请求负载方面来看，虽然对NameNode单机的优化可以对压力进行缓解，但是从长远来看，需要对NameNode进行分布式改造。

在NameNode的分布式改造中，我们研究了Ceph、GFS、Lustre和Hadoop社区对分布式元数据管理的实现和思路。

Ceph文件系统的分布式元数据管理实现

● Ceph是加州大学圣克鲁兹分校的Sage Weil攻读博士时开发的系统，并且也以Ceph完成了自己的博士论文。2007年博士毕业后，Sage Weil依旧全职投入到Ceph的开发中。2010年3月19日，Linus将Ceph Client合入了Linux 2.6.34的内核，从此Ceph开始吸引了大批人的关注。

● Ceph与其他分布式文件系统的重要区别就在于元数据的分布式管理上。在过去，包括GFS和HDFS在内，通过将数据分散存储到多个节点上，使得数据的存取得以线性扩展。但是，始终影响整个系统扩展能力的是无法充分地对元数据进行分布式管理。为了使得系统在大规模下依旧扩展，在元数据管理这方面产生了两种道路：一种是废弃文件系统的概念，就是不提供传统的文件目录树的命名空间，比如Amazon的S3，只提供了简单的两层命名空间；另一种是在提供文件系统目录树的前提下，继续探索各种扩展技术。在分布式元数据管理这方面，S3通过简化命名空间管理极大地提高了系统的可扩展行，但是对于很多应用和开发者而言，他们更熟悉文件系统目录树的命名空间。Ceph融合了这两种道路，既提供了一个高度可扩展的分布式对象存储系统（类似S3），也在这个对象存储系统之上提供了一个高度可扩展的分布式文件系统。

● Ceph文件系统的三个特别之处：第一，动态的分布式元数据管理；第二，提供了一个自治的、可靠的、可扩展的对象存储层（RAD-

OS）；第三，为RADOS提供了一个层次数据分发函数（CRUSH）。

● Ceph设计独特之处就在于对象存储层的独立。首先，RADOS利用CRUSH函数，根据集群的组成拓扑情况和ObjectID，可以计算出Object的实际存储的多个副本位置，这样即达到了不经过Master层，可以直接进行Object的读写；其次，RADOS内部利用一个P2P的协议来维护数据的复制、错误发现和恢复、数据迁移，提供严格的一致性。

● 在RADOS上，可以建立多种Namespace，这里的Namespace提供对Object的管理或者索引。比如可以提供类似层次文件系统，提供类似于Amazon S3的Namespace。

● Ceph基于RADOS实现了一个层次文件系统，这个文件系统采用了分布式元数据管理的方式。

● 元数据管理有多个MDS组成，MDS只是文件系统元数据的内存Cache，每一个MDS的元数据和日志都存储在RADOS上，以便于进行Failover操作。

● 文件Inode、Direntry（文件名和Inode号的对应关系）都存储在相应的目录Inode中，每一个目录Inode存储在一个RADOS的Object上。Inode和Direntry都存储在目录Inode中的好处是访问目录中各个文件的效率会高很多，因为应用的访问模式都具有目录局部性。坏处是实现Hardlink的困难，在Ceph中，考虑到一个文件有多个Hardlink的比较少，所以提供了一个Anchor Table来存储Inode到文件名（Primary Link）的映射，其他Hardlink被称作Remote Link。

● Large Journal。MDS通过写入一个较大的Journal来减小对元数据文件的更新频度。这样MDS可以定期地更新存储到RADOS上的元数据，在更新周期内，通过Journal来保证元数据的持久化。

● Dynamic Subtree Partition。不同于Lustre，Ceph是基于子树进行分割。即每个目录有不同的MDS负责，如果这个MDS负责这个目录，那么这个目录的父目录和祖先目录都需要Copy一份过来。当前MDS负责的目录称作Authority，同一个目录在其他MDS上的副本称作Replica。MDS通过之间的协议来维护Authority和Replica的一致性，并且根据目录的请求负载和其他性能参数可以对子树进行动态迁移。如果一个目录负载很高，会通过复制来将压力分散到多个节点上；如果一个目录非常大，也可以通过将目录进行分割的方式，并且将各个Directory Fragment分散到多个MDS来解决超大目录问题。

总结：提供一个统一的层次文件系统命名空间；支持元数据分布到多个元数据节点；支持Failover；支持根据目录的访问负载对元数据进行动态重分配。

Lustre的分布式元数据管理实现

● Lustre的每个元数据管理节点称作MDS。所有MDS组成一个Clustered MDS。

● 所有的MDS共享一个持久化存储层。MDS只是内存Cache，依赖这个共享的持久化层也方便了MDS的Failover机制，提高Availabililty。

● 每个MDS负责多个Inode Group组。这里的一个Inode Group基本等同于一个目录，因为在某个目录较大时，会将一个目录分割成多个Inode Group。

● 至于一个Inode Group由哪一个MDS进行加载，有多种算法，比如HASH、Round-Robin、随机选取、根据负载进行选取等。这个可以借由一个叫做Resource Manager的服务来维护Inode Group到MDS的映射关系。

● 由于每个MDS只是维护各个独立的Inode Group，要想获得一个完整的层次结构，需要对多个MDS进行查询。比如客户端访问一个文件，为了判断权限，可能需要遍历多个MDS。为了加快这种查询，Lustre在客户端做了较多缓存的工作，缓存相应的层次结构，以便下次访问相同的层次结构时可以在本地缓存中获得。这个机制之所以奏效是由于大部分客户端访问都有一定的局部性。但是缓存的有效性管理难度还是比较大的，在Lustre中实现了一个分布式锁服务。

● MDS也是通过记录日志的方式来保证元数据写入的Durable。Failover后会通过日志回放的模式来在另一个MDS上进行恢复。

● Rename等类似操作的实现可能需要跨越多个MDS来完成。

总结：提供一个统一的层次文件系统命名空间；支持元数据分布到多个元数据节点；支持Failover；不支持根据目录的访问负载对元数据进行动态重分配。

Google文件系统在元数据管理方面的实现和下一代GFS的可能实现方案

● GFS从一开始提供的就是平坦化的命名空间，而不是层次命名空间。这个命名空间的维护如同Hadoop的HDFS一样，也是在单机进行维护。

● GFS发展过程中，也遇到了类似Hadoop的NameNode的扩展问题，Master节点出现内存压力和性能瓶颈。其中原因包括集群规模的增大，大量小文件的出现。

● GFS给出的临时方案是；对于各个部门或者产品组自己搭建一个GFS，而多个GFS集群共享同一个物理集群，即有多个Namespace，应用自己维护应该访问的GFS集群的地址；对于那些需要跨越多个集群进行存储的应用，Google开发了一个叫做Namespace Tool的工具，这个工具使得访问底层的多个GFS集群对应用透明。

● BigTable的出现在一定程度上缓解了小文件带来的问题严重性，但是BigTable不是解决Master单点的理想解决方案。其原因包括：设计目标不同，Bigtable复杂化了一个文件系统的简单性；BigTable的空间回收利用的是GC机制，这个在大压力下可能造成空间使用效率问题。

● 基于下面的需求考虑，Google提出了GFS II：内存的压力，越来越多的文件/小文件，Master的内存受限；元数据的处理随着集群规模的增大，每秒钟需要处理的请求数越来越多；Availabilty，Master是个单点，而GFS面向交互式应用后，对Availability的要求提高；Latency，GFS面向交互式应用后，对latency的要求变高，一方面需要Availability的提高，另一方面需要考虑分布式化，尽量降低单个请求的处理时间。GFS II的设计目标是可以扩充到上百个Master。

● GFS II的设计猜想：纯内存的BigTable来存放元数据；数据块的维护可能类似Ceph的RADOS，不需要Master节点做过多的管理。

总结: 提供一个统一的平坦化的文件系统命名空间；支持元数据分布到多个元数据节点；支持Failover；可能不支持根据目录的访问负载对元数据进行动态重分配。

Hadoop社区的分布式元数据管理的改进计划

● 采用类似GFS I中的方案。支持多个Namespace（多个Namenode），主要还是有应用自己来维护自己所要使用的Namespace，Namespace之间的界限对应用不透明。

● 主要设计：DataNode在所有NameNode间共享，每一个NameNode有自己的Block集合，被称为Block Pool；一个Block属于一个Block Pool和NameNode；Block用<PoolID><BlockID>来唯一标识；DataNode端，将不同Pool的物理块存放在各自单独的目录下。

● 主要实现：不改变当前块管理和命名空间管理的实现，NameNode管理自己的Block Pool和Namespace；DataNode向多个NameNode注册，但仅将Block汇报给其所属的Block Pool/NameNode。

总结：本质上只是一组共享了DataNode的、独立的NameNode；元数据在NameNode之间的分布对用户不透明，换句话说，用户看到的就像是多个HDFS集群；解决了DataNode空间利用不均的问题，但是NameNode的元数据仍然可能不均。

总结

Ceph完成的功能最完善，解决了元数据的Scalability、Avalability和动态负载均衡。Ceph的实现总体感觉复杂度较高，估计相应未来的问题也会比较多。

类似Ceph的块存储层和Namespace分离，可以在一个块存储层上灵活地支持多种Namespace。估计GFS II也会采用类似Ceph这种块存储管理和Namespace管理分开的实现思路。

Namespace和底层存储管理分开后，可以实现为各种类型，可以采用各种分布式方式。

为了提供一个更友好的使用接口，还得需要提供一个统一的命名空间，而不是提供多个命名空间。

作者简介 | 马如悦

百度基础架构部高级工程师，自2007年加入百度，一直从事分布式存储系统和分布式计算系统的设计和开发工作。对Hadoop有较深入的研究，一直积极活动在Hadoop开源社区。

■ 责任编辑：董世晓（dongsx@csdn.net）

详图实证：再谈JavaScript的语源问题

■ 文 / 周爱民

本文通过大量的图片资料，证实了几个鲜为人知的JavaScript的语源问题。

在JavaScript中有两个错误观点。第一个错误观点是“JavaScript在语源上继承自Cmm”。该观点主要来自以下途径（部分）。

- 2002年10月7日《Wired Magazine（连线杂志）》的一份名为“Mother Tongues”的图。
- O’Reilly公布的“The History of Programming Languages”图。
- Levenez.com公布的“Computer Language History”。

第二个错误观点，即“Nombas公司的Espresso Pages（浓咖啡版网页）及其内置的脚本（CEnvi，Cmm语言的开发环境）是因特网上首次使用的脚本语言。该观点主要来自以下方面。

- 在Nombas网站中关于Cmm、CEniv以及ScriptEase等技术与产品的一篇介绍文字；
- Brent Noorda先生（Nombas公司总裁）关于发布Espresso Pages的新闻组消息。

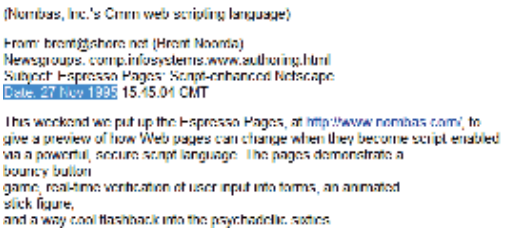
这些明显是不太靠谱的一面之词。但进一步援引这些内容的结果是：错误的观点被一再重述，似乎已经快要变成事实了，例如《JavaScript高级程序设计》一书已经白底黑字地将这些内容记述在了“JavaScript简史”中。

本文将简明而完整地厘清上述观点。

真相：Netscape早在Espresso Pages发布前就实现了主要的Web开发特性

首先，这些问题的一部分相当容易说清楚，例如“是不是Espresso Pages首次在Web中使用了脚本语言”。因为关于Espresso Pages的这个新闻组消息还有着明确的时间（1995.11.27）[1]，如图1所示。

那么，Netscape Navigator 2.0中对脚本的支持又是如何出现的呢？考察NN2此前发布的各个Beta版就可以发现[2]：至迟到NN2 Beta2，网


(Nombas, Inc.'s Cmm web scripting language)

From: brent@shore.net (Brent Noorda)
Newsgroups: comp.infosystems.www.authoring.html
Subject: Espresso Pages: Script-enhanced Netscape
Date: 27 Nov 1995 15:45:04 GMT

This weekend we put up the Espresso Pages, at http://www.nombas.com/, to give a preview of how Web pages can change when they become script enabled via a powerful, secure script language. The pages demonstrate a bouncy button
game, real-time verification of user input into forms, an animated stick figure,
and a way cool flashback into the psychedelic sixties

图1 Espresso Pages新闻组消息

页内嵌脚本的概念就相当清晰，并且实现完整了。我们先看看在1995.10.10发布的NN2 Beta1版本，它能做什么呢？它其实已经具备了标签内联脚本功能，如图2所示。

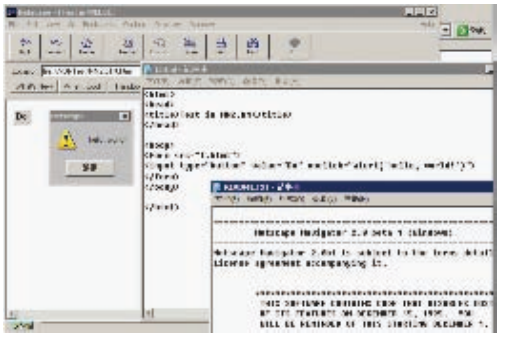


图2 NN2 Beta1已具备标签内联脚本功能

相对完整的脚本特性，例如<script>标签与function函数等，则要等到NN beta2才支持，其发布时间为1995.11.04，如图3所示。

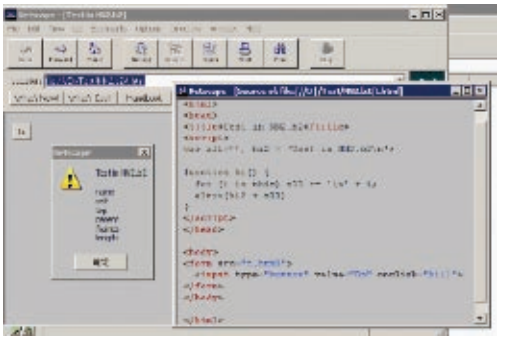


图3 NN beta2支持相对完整的脚本特性

图3的示例说明Beta2中的脚本已经开始支持在函数内通过this引用来得到window对象，而frames、self等成员在这时的window对象中就已经存在，这些事实上就是最原始的DOM。

所以尽管Nombas抢在Netscape Navigator 2.0正式发布（1996.01.23，JavaScript 1.0随该版本发布）之前，公开了在NN2上嵌入第三方脚本引擎的方式来实现的Espresso Pages，但与我们今天讨论的Web开发相关的一切，在这之前、在NN Beta2中就已成为事实。而Netscape Navigator Beta2的发布时间比Espresso Pages早了近一个月。

《JavaScript高级程序设计》一书中不但认为Espresso Pages是因特网上“首次使用脚本语言的标志”，还继续讲道：

相信当初的Nombas公司不太可能意识到，他们这种网页中嵌入脚本的想法会在很大程度上左右未来因特网的发展。

这段“文字游戏”很容易让读者将“网页嵌入脚本”的设想归功于Nombas以及具体的脚本语言Cmm。但如上所述，这种设想并非源自Espresso Pages。这些想法的确左右了因特网的发展，但是没有任何证据显示Nombas与这一起源存在关系。

真相：Cmm与JavaScript完全无关，而ScriptEase不过是追随者

Cmm的确是一种脚本语言，比JavaScript出现得早很多。但它并不是嵌入式的，需要一个名为CEniv的可执行程序，来运行这种扩展名为.cmm的脚本代码文件。因为Cmm一开始并没有设计为嵌入式语言，所以这些称为Espresso Pages的东西其实是一些可供下载的Demos，使用者需要安装CEnvi的某个版本，并通过一些配置来使Demo能在NN2 beta中使用。原文是：

but for now these demos use our CEnvi for Windows application as a helper. Instructions are on the page for how to configure, including a download of a demo of the Cmm interpreter.

现在已经再也找不到这些下载和操作指南了。我们这里并不想考察Espresso Pages的效果，只想说清楚他当时所使用的Cmm语言与JavaScript语言有什么关系。因为当时JavaScript正在开发之中，而Cmm已出现了三年之久，那么JavaScript的语言设计是否参考了Cmm呢？

目前可以下载到的CEnvi包括以下版本：

CEnvi 1.0	– 1993.08.09
CEnvi 1.008	– 1993.12.21
CEnvi 1.009 for DOS/Win/OS2	– 1994.11.29
CEnvi 2.0 for DOS/Win/OS2	– 1995.03.29
CEnvi 2.10 for DOS/Win/NT/OS2	– 1995.10.17
CEnvi 2.11 for DOS	– 1996.02.20

在NN2之前的是CEnvi 2.0，NN2正式发布之后则有CEnvi 2.11。我们只需要考察这其中Cmm语言的特性及其进化的路线，就可以了解这一阶段中二者可能存在的相互影响。根据Cmm的语言手册[3]，其主要的发展为：

- 1.0~2.0：添加了字符串中的转义字符(Escape Sequences for Characters)；
- 2.0~2.1：无变化；
- 2.1~2.11：无变化。

也就是说，Cmm语言从1993年随CEnvi发布直到1996年2月，下面的语言特性并没有任何实质性的变化：

- 与C语言相似的运算符（包括位运算、布尔运符和三元“?:”运算符等）、注释、字符串表达方式；
- 有Byte、Integer、Float等数据类型，不需要预声明变量，变量在第一次赋值时创建并关联类型；
- 有数组与字符串类型，数组不需要预定义长度或元素类型；字符串是字符类型的数组，可以通过数组下标存取；通过“...”方式，或用{...}声明一系列单个字符的方式来声明字符串；
- 有结构（Structures）数据类型，用“.”字符存取其字段；
- 有if、while、do、switch、for、GOTO等语句；
- 可以声明函数，但不需要用function关键字开始，也不需要声明参数类型，参数个数必须是确定的；
- 有#include与#define等预声明语句；在任意函数之外的代码为初始化代码，其声明的变量为全局变量；在初始化代码之后执行的第一个函数为main(ArgCount, ArgStrings)。

可见Cmm语言的多数概念来自于C语言，而与JavaScript所体现的面向对象特性毫无关系。例如Cmm的数组是一个独立的数据类型，取得数组的长度应使用GetArraySpan(aArr)函数，而不是取数组这个对象的一个属性aArr.length。通过对Cmm语言特性的详细分析可见：即使JavaScript与Cmm存有一些相似的

部分，那么最多也是其借鉴自C语言的表达式运算、语句语法等，而这些并不足以成为“JavaScript语源自Cmm”的证据。

而在这个话题中的另一个角色，是名为“ScriptEase”的脚本语言，它也是最早符合ECMAScript 262标准的语言之一——简单地说，它也是一个JavaScript语言。而且，Nombas公司正是因此将Cmm与JavaScript挂上关系，因为它声称：ScriptEase就是Cmm，只不过是换了个名字。真的只是换了一个名字这样简单吗？

此前我们说过，Cmm的最后一个可见的发布包含在CEnvi 2.11中，发布时间是1996.02.20，其语言特性与上述列表一致（没有变化）。这个语言是什么时候更名的呢？答案是：晚于1997.02.06。因为在1997年4~6月间，Nombas公司发布了CEnvi的3.0版本，在这个版本中首次使用到ScriptEase这个名字，其中的名为HISTORY的文件——该文件被打包的时间（令人惊奇地）是1997.02.06——图4描述了这次更名的缘由。

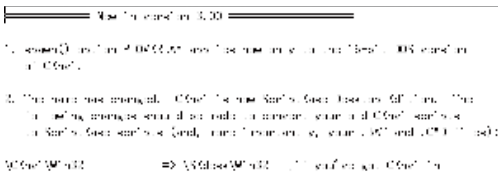


图4 ScriptEase名字的由来

那么在这个版本中的ScriptEase又具有怎样的语言特性呢？答案是：与Cmm 1.0~2.11相比仍然没有变化。对比ScriptEase 3.0发布包中的语言手册《The ScriptEase Language》与Cmm 2.11中对应文档可知：数组、字符串、函数、GOTO语句等一切，仍跟Cmm中一模一样。

但这个原来叫Cmm的语言摇身一变，改名叫ScriptEase了。现在，请看图5中的代码（语言手册对比，左侧为ScriptEase 3.0 - 1997.04.15文档，右侧为Cmm 2.11 – 1996.02.20文档）。

相信任何一个看过JavaScript代码的人都会明白，这个名为Cmm/ScriptEase的语言，所采用的函数声明的语法绝非JavaScript语法，也绝对不可能符合ECMAScript所定义的规范。而此时是1997年，JavaScript 已经发布了两年，ECMAScript标准即将发布（1997.06）。虽然现在的Cmm已经更名为ScriptEase，但在其语言手册中仍然找不到Object这样的词汇，代码上仍然使用着最初的函数声明语法。

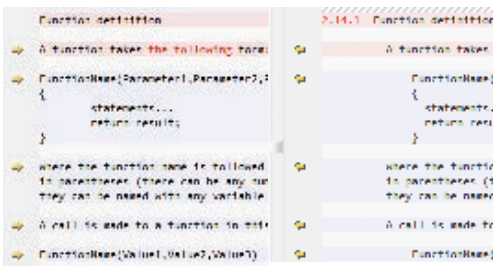
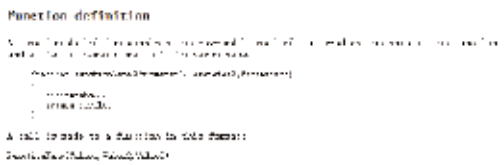
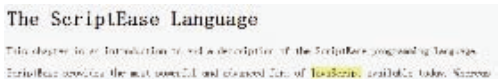


图5 ScriptEase 3.0与Cmm 2.11语言手册对比

那么，这样的一门语言又是何时摇身一变，以至于被误认为“JavaScript的语源”的呢？这还要推进到半年之后，也就是1997年12月。ScriptEase发布了4.0版，在目前可以找到的一个名为“ScriptEase:WebServer Edition 4.01 – 1997.12.08”版本中（准确地说，我并没有找到3.0至4.0的全部发布过程），终于有了一份手册，开始采用如下的语法：



正是在这份ScriptEase官方手册中，写道：



这一切还需要解释吗？以下两个言论：

- ScriptEase在从3.0到4.0的过程中，采用了ECMAScript标准，进而实现了JavaScript的一个版本；
- JavaScript语源自Cmm，继承了后者语言特性。

可见后者是何等站不住脚？！然而，它正是这样地画在了O’Reilly的图册里[4]，如图6。

真相：Nombas公司的广告做得太好

大概是2001年前后，Nombas官方网页对其ScriptEase的介绍中，使用着一份名为“History of ScriptEase and JavaScript”的文档[5]，但这个网页使用着与该文档标题完全不同的TITLE图片，如图7所示。

这篇文档序言写得中规中矩。大意是说1992—1993年间，Nombas开发了Cmm语言，讲到了该语言是“embeddable scripting language”，如前所述，这时的Cmm并没有嵌入式的特性。不过，从后来Nombas总是同时发布CEniv的不同操

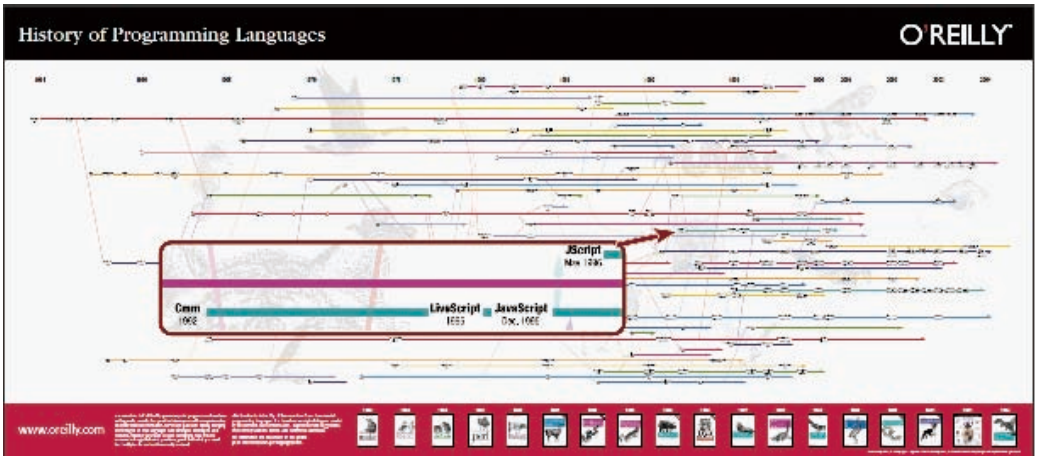


图6　O'Reilly中登出了JavaScript语源自Cmm

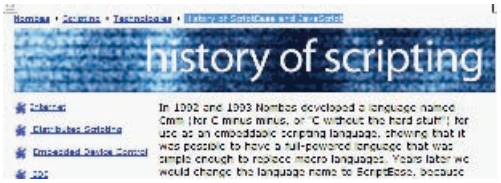


图7　Nombas官网对ScriptEase的介绍

作系统平台版本来看，Cmm语言是采用了跨平台的语言（例如C）编译的。但“Cmm随CEnvi发布不同平台版本”并不能作为它“具有嵌入特性”的充分证据。再退一步来看，即使Cmm具备了嵌入式语言特性，可作为独立模块来嵌入到CEnvi发布，也并不表明它可以嵌入到后来的Netscape Navigator中，因为后者在当时是闭源的，没有可能让CEnvi来“嵌入”。然而这一点，在《JavaScript高级程序设计》中被写成：

When the popularity of Netscape Navigator started peaking, Nombas developed a version of CEnvi that could be embedded into web pages.

可惜这并非事实。Nombas的原文是：

When Netscape's first commercial browsers were released we made a version of CEnvi that could handle short scripts embedded within web pages.

Nombas并没有在这里指明“first commercial browsers”是Netscape的哪一个具体版本。因此这段文字很容易让人误解为NN 1.0发布的时候，CEnvi就已经可以实现网页内嵌脚本了。而事实上，所谓的“CEnvi定制版本”根本就没有出现过，即使曾经有过，也不过是Nombas在NN 2.0 beta中玩的一些小手脚。进一步地说，这时的Cmm、CEnvi以及后文中的“Espresso Pages”，都只是Nombas在NN 2.0 Beta发布阶段的一些技术尝试而已。

然而，这篇Nombas的原文又继续谈论“网页内嵌脚本”的一些特点。例如：

By embedded scripts within the page we allowed the client side to handle processing, rather than making all dynamic interaction happen on the server. This brought immediate client-side interaction with the user.

这些特性描述也就仅仅是对这类技术的一个概述而已，并不表明什么有“原创”、“独创”或“初创”的意义（注意这里使用了“句号”）。但是接下来，Nombas耍了一个小花招，在文章中说：

The advantages of client-side handling were made obvious by Nombas' "Espresso Pages", and Netscape soon began work on their own version, which they called LiveScript, and then renamed to JavaScript just before its final release.

注意，“Netscape很快在它自己的版本上开始工作”之前的“and”，这里可以理解为“接下来”，或者“并且”，或者仅仅是语气上的连贯，但无论如何，这此前一个“逗号”，以表明“their own version（他们自己的版本）”与前面所述的一切存在“必然的”关系。然而如本文此前所述的：

- “Espresso Pages”中网页内嵌脚本的想法，在NN Beta2中就已经成熟了，而后者早前了一个多月就发布了，谈不上受到了“Espresso Pages”的影响。

同样，正是由于这个“And”和逗号，使得LiveScript与后面的JavaScript“变成了”Cmm的一个后来者，被错误地解读成了“存在语源

上的关系”。而本文此前的分析说明：

- Cmm受到了JavaScript的影响：ScriptEase在从3.0到4.0的过程中，采用了ECMAScript标准，进而实现了JavaScript的一个版本。

正是Nombas的原文中的措辞，不可避免地让人认为LiveScript是基于“Espresso Pages”思想的（或受到其影响的）一个版本，进而发展成为JavaScript。但是事实上：

- JavaScript最早被称为Mocha（魔卡），这是项目的代码名。这个名字一直用到NN2.0 beta 2发布之前（1995.11.04）——包括在beta 1中弹出的错误框上，还可以看到Mocha的名字。
- NN2.0 beta2发布的时候使用了LiveScript这个内部名称。
- 随后Netscape与SUN共同发布声明（1995.12.04），正式启用JavaScript这个名字。直到NN2.0 beta4发布时（1995.12.20），其文档与软件界面中才开始统一使用JavaScript这个名字。

这一切，与“Espresso Pages”是否出现，有什么样的关系吗？——毫无关联！

最后的定论，Brendan Eich的澄清

在Wiki Talk中保留着Brendan Eich的一段对话[6]，足以为Cmm与JavaScript之间的关系做最终的澄清。

Hello. I first met Brent Noorda in late 1996, when Netscape brought JavaScript to ECMA for standardization. I had never heard of NOMBAS or its products before then. When I created JS in May 1995 (in about ten days for the core language implementation; the rest of that year was consumed by the DOM and browser embedding work), my influences were awk, C, HyperTalk, and Self, combined with management orders to “make it look like Java.”

其一是与Cmm/Nombas的关系：

我第一次见到Brent Noorda是在1996年底，在此之前我从未听说过NOMBAS或它的产品。

其二是JS语言的创建过程和大概的时间：

在1995年5月开始创建JS的时候（大约10天用于核心语言实现，其他的时间主要是用在DOM以及浏览器嵌入方面）。

其三是关于JS的语言的最终结论：

我的影响主要来自于awk、C、HyperTalk和Self，以及主管们所要求的“使它看起来像Java一点”。

关于第三点，JavaScript“像”Java，而不是“语源自”Java，可以由《JavaScript Working Document 1.0》（1996.01.22），以及其原始的、Netscape公司的官方文档《JavaScript Authoring Guide》文档中的原文[7]：

The JavaScript language resembles java, …

以佐证。JavaScript在语法上与Java、Awk和Perl都有一定“形似”，这可以由《JavaScript Language Specification V1.1》（1996.11.18，Brendan Eich参与编订）文档中的原文[8]：

Borrows most of its syntax from Java, but also inherites from Awk and Perl

以佐证。最后，关于Brendan Eich所言：

my influences were awk, C, HyperTalk, and Self

其中的关键语言特性——原型继承、动态类型和动态数据绑定等，并未出现在Netscape Navigator 2.0及其中的JavaScript 1.0中，因此这里不再详究。

参考资料

[1] http://embedded.eecs.berkeley.edu/Respep/Research/dds/Hyper-RNweb/0052.html
[2] 关于Netscape Navigator 2.0，其Beta版的发布时间采用安装包文件的打包时间，其正式版的发布时间采用官方公告时间。
[3] 关于Cmm语言的特性与版本变更的相关结论，是通过对照各个CEnvi版本中的Cmm语言手册（CEnvi Demo Manual, Chapter 2: Cmm Language Tutorial）而得出的。
[4] http://oreilly.com/news/languageposter_0504.html
[5] http://web.archive.org/web/20040618115018/http://www.nombas.com/us/scripting/history.htm
[6] http://en.wikipedia.org/wiki/Talk:ECMAScript#Origins_of_LiveScript
[7] http://tecfa.unige.ch/guides/js/js10.pdf
[8] http://www.planetpdf.com/codecuts/pdfs/tutorial/jsspec.pdf

作者简介 | 周爱民

国内软件开发界资深软件工程师，技术作家。有10余年的软件开发、项目管理、团队建设的经验，现任支付宝公司业务架构师。著有《JavaScript语言精髓与编程实践》等。

责任编辑：董世晓（dongsx@csdn.net）

算法之道——形而之上谓之道

■ 文 / 邹恒明

本文以最小生成树问题为例，全方位展示了算法背后的逻辑和战略。

1966年3月的一天，美国加州大学洛杉矶分校的Andrew J. Viterbi教授在给研究生讲解缠绕编码的时序译码算法SDCD。但不管他如何讲解，学生就是听不明白。思来想去，Viterbi觉得学生不能理解的原因是该算法的证明过于复杂。于是他开始考虑如何简化这个证明。在经历了持久的烦躁和困惑后，他灵感顿现：需要简化的不是算法的证明，而是算法本身。于是Viterbi对SDCD算法进行了少许修改，提出了基于Trellis的概率译码算法。这个算法就是后来著名的CDMA技术的基石。Viterbi也因此而身价暴涨（创立了高通公司，赚取了数十亿美元）。

一种新算法引来革命性的技术和财富的暴涨，算法的作用不可谓不大。但理解算法、改变人生，或者以算法的思维来进行思考，却对很多人来说是镜中花、水中月，难以触摸。

从广义上定义，算法就是求解问题的步骤（指令）。由于计算机的程序是一条指令接一条指令（步骤）地进行，它实际上就是算法的逐步展开。因此，算法弥漫在所有的软件程序里，堪称计算机的灵魂。

理解灵魂当然不是件容易的事情。除了高度抽象外，算法背后的逻辑也非常缠绕。在看过一个问题的算法解答后，人们感到的不一定是轻松，反而可能是困惑。这些算法是如何被发现或发明的呢？这些问题的解答者是如何想到特定的算法呢？在某些情况下，人们还不一定问得出这样的问题，因为对算法本身可能还没有看懂。其实，深刻掌握算法的人并不多见。尽管很多人会在各种场合指点江山、激扬文字，俨然一副大师的架势，但他们对算法的理解可能十分肤浅。很多经常需要使用算法的人，也不过停留在自发，而不是自觉的阶段：对于见过的问题或者与见过的问题类似的问题知道如何作答，而对于那些与见过的问题没有相似性或相似性很少或相似性不容易看出的新问题就一筹莫展了。

而各种算法书籍中的内容堆积、枯燥陈述、逻辑凌乱甚至理解错误等现象则加剧了人们对算法的畏惧。

市场上的算法书籍琳琅满目，但存在共同的问题：讲述一大堆问题，罗列诸多算法，但从根本上却是就事论事；各种算法设计或分析战略之间没有什么逻辑递进或层次关系，算法各种战略的顺序在安排上非常随意，与它们之间存在的因果关联不相符合。比如，这些书籍在讲动态规划、静态规划、贪婪选择、近似算法等时，没有考虑到不同战略之间的逻辑递进关系，只是随意安排章节，用一些具体问题来讲解这些战略而已。结果就是没有逻辑主线贯通，读起来费力、分散，不能形成有机的整体。看这些书的唯一收获是获得各种具体问题的解答，但在见到一个新问题时，对到底应该使用何种设计战略和分析战略，或者应该以何种顺序来尝试各种战略却模糊不清甚至不得章法。此外，这些算法书对算法战略并没有进行提炼，而是凌乱地分散在各种问题的具体解答中，形不成一种高度，形散神更散，无法系统性地训练读者的算法思维。

这些书作为工具书查阅倒也可以，但作为训练算法思维的读物或者教材显然力有不逮。

那么如何培养算法思维呢？答案就是算法背后的逻辑。不同的算法战略看似不同，实则一脉相承，甚至从更高的层次上看就是同一种思维。所有的算法战略如分而治之、动态规划、贪婪选择、随机化、近似算法等只不过是同一思维的不同方面而已！它们之间存在逻辑和效率的递进关系。明白了这一点，对算法的把握就会达到一个新的境界。我们用众所周知的最小生成树问题为例来加以说明。

最小生成树问题的定义如下。

给定输入为：带权重的连通无向图$G=(V,E)$，每条边的权重为w_i。

要求输出：一棵连接所有节点的树T，并且$w(T)=\sum_{(u,v)\in T}w(u,v)$为最小。

例如，图1里面的粗线条组成了该图的一棵最小生成树。

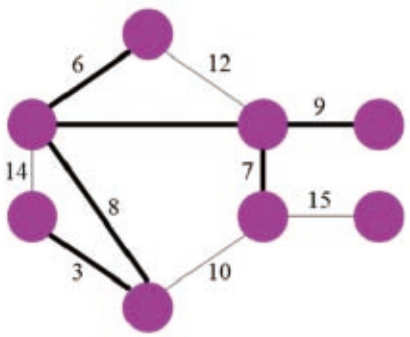


图1　最小生成树问题，图中粗线组成一棵最小生成树（图片来源：《算法之道——从无有到无穷》）

我们是如何获得这棵最小生成树的呢？或者最小生成树问题该用什么方法来解决呢？

最简单的办法当然是暴力战略，即将所有的生成树找出来，计算它们的权重，取出最小的树即可。但此种战略的成本高昂，其数量级为$C_E^{V-1}\Theta$，这里E代表图中的边的条数，V代表图的节点数量。而这是一个难以令人兴奋起来的阶乘级$\Theta(E!)$。显然，我们需要对算法进行改进。那么如何改进呢？

在面临复杂问题时，人类通常选择将问题简化，即将复杂的大问题分解为简单的小问题。在解决了小问题后，再将小问题的解合并为大问题的解。这就是所谓的“分而治之”。由于小问题比大问题更易解决，分治就成了上策，并演化为算法设计的基础战略。对最小生成树问题来说，就是将整个图分解为两（也可以是其他数量）个尺寸（节点数）相等或相近的子图，分别在这两个子图上寻找最小生成树，然后将寻找出的两个最小生成树合并起来即可。显然，分解的成本为线性，但合并的成本是$\Theta(V^2)$，这样我们得到分治算法的成本递归式为$T(V)=C_V^{V/2}T(V/2)+\Theta(V^2C_V^{V/2})$。按照大师解法，该递归式的解为$\Theta(V^2C_V^{V/2})$。

这是最优解法吗？仔细分析上述分治算法的成本构成可以发现，在分解$C_V^{V/2}$个子问题时会出现很多重复的下级子问题，重复解决这些相同的子问题显然不是明智之举。改进的办法就是将重复的子问题解决一次，然后将结果存起来供以后使用，这就是动态规划战略。采用此种战略后，最保守也可以将成本降低至$\Theta(\sum_{i=1}^{V}C_V^i)=\Theta(2^V)$（实际上，在略为优化后可将成本降低到$\Theta(\sum_{i=1}^{\lfloor \lg V\rfloor}C_V^{V/2^i})$）。

但这是最优解法吗？其实还不是。仔细分析可以发现，由于构建的是最小生成树，我们可以依次选择图G里面最小的边作为最小生成树的边，条件是新选择的边不与已经选择的边构成环路，直到有$V-1$条边入选为止。而这正是贪婪选择战略。由于将所有的边排序的时间复杂性为$O(E\lg E)=O(E\lg V)$，检查一条边被选中后是否与前面选择的边形成环路的时间复杂性为线性，因此整个算法的时间成本为$O(E\lg V)$。如果$E\ll V^2$，此算法的效率将高于前面讨论的标准分治战略的效率。如果再使用改进的数据结构来支持贪婪选择战略，则该时间复杂性可以降低到$O(E+V\lg V)$（使用斐波拉契堆的摊销时间）。

但贪婪选择战略还不是最优战略。仔细分析上面最小生成树的构造算法，我们发现它的成本在于边的选择（排序是为选择做的准备），而构造最小生成树本身的成本只有$E+V$。我们为什么要花多于构造本身成本的成本来构造最小生成树呢？假如我们知道哪些边属于一棵最小生成树，则构造起来就不费力气了。但要想知道哪些边属于最小生成树，不是需要与其他边进行比对吗？也许我们并不需要。我们可以用一个随机数来告诉我们一条边是否属于最小生成树。准确地说，我们用抛硬币来决定一条边是否应该被纳入到最小生成树里。这样就可以将时间成本降低到线性$O(E+V)$。这就是随机化战略。

这样，随着算法战略的递进，寻找最小生成树的成本不断降低，且在随机化战略达到线性的最低点（表1）！

表1　算法战略递进中的最小生成树构建成本

算法战略	时间成本
暴力战略	$\Theta(E!)$
标准分治战略	$\Theta(V^2C_V^{V/2})$
动态规划战略	$\Theta(2^V)$（优化情况下：$\Theta(\sum_{i=1}^{\lfloor \lg V\rfloor}C_V^{V/2^i})$）
贪婪选择战略	$O(E\lg V)$（斐波拉契堆：$O(E+V\lg V)$摊销时间）
随机化战略	$O(E+V)$

不过，细心的读者可能会产生诸多问题。例如，贪婪选择战略下，为什么会想到使用斐波拉契堆呢？斐波拉契堆的摊销分析结果

$O(E+V\lg V)$是如何得出的？在随机化战略下，如何保证所选的边确实是最小生成树的边呢？另外，表1中的贪婪选择战略似乎与随机化战略不相上下：$O(E+V\lg V)$和$O(E+V)$不是一个数量级吗（E通常是大于V的）？怎么说随着算法战略的递进，成本不断降低呢？

在贪婪选择战略下，我们每次选择权重最小的边加入到一棵初始为空的最小生成树里，被加入的边不能与已加入的边形成环路。在这种战略下，算法的最大成本在每次选择最小的边上。而堆是支持此种操作的最佳数据结构。但对普通的二叉堆来说，每次删除堆顶（我们当然使用最小堆）元素时，需要对堆进行调整以保持堆的属性，从而为后续的取最小边操作打下基础。由于此种调整的时间为对数级，使得整个最小生成树的操作成本为$O(E\lg V)$。但如果我们改变思维，取消二叉堆要求每个节点度数不超过2的限制，则调整堆的操作成本将降低到常数级。在这种情况下，每次删除堆顶元素后，直接将元素较大的子堆挂在元素较小的子堆下即可。这样，最小生成树的选边操作的总成本似乎为$O(E)$。加上降距操作的$V\lg V$成本，整个最小生成树的成本就是$O(E+V\lg V)$。

等等，这似乎有一个问题：前面所述的堆调整操作为常数级的前提是删除堆顶元素后出现的子堆个数为常数。否则，在一大堆的子堆中间选出最小的元素（从而将别的子堆挂在其下）则可能不是常数时间。因此，问题的关键在于确保删除堆顶元素所产生的子堆个数为常数或者有限。而这种思路就导致了斐波拉契堆的出现。事实上，斐波拉契堆比这更进一步：合并操作也不是马上进行，而是留下这些子堆，在子堆数超出一定的限制时才进行合并操作。此外，出现度数相同的堆（堆顶元素具有同样子节点数）时也进行合并操作：将度数相同的堆合并成新的堆。由此，我们可以得出下面的斐波拉契堆的定义。

- 斐波拉契堆由一组普通的堆（非二叉堆）构成。
- 度为k的节点的任意一棵子树的规模最大为F_{k+2}（每个节点的子节点数不能超过$O(\log n)$）。
- 所有子堆的度数都不相同。

图2给出的是一个斐波拉契堆。

在斐波拉契堆下，删除一个堆顶所产生的子堆个数不会超过$O(\log n)$。因此，在留下来的子

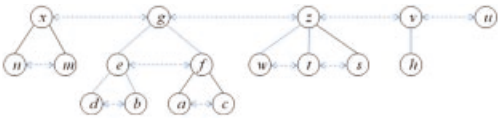


图2　斐波拉契堆结构示意图（图片来源：《数据结构之弦》）

堆里面寻找最小元素的时间成本也不会超过$O(\log n)$。但这样似乎得不到我们想要的$O(E)$的时间。不过，仔细分析发现，在斐波拉契堆的限制下，不可能每个节点的子节点数都是$O(\log n)$。事实上，绝大部分节点的子节点数都很少。这样少数几个节点的高度数导致的高成本可以摊薄到大量的低度数节点的低操作成本上，从而将整个最小生成树的操作成本降低到每次选边操作为常数成本的境界。这就是摊销分析的中心思想。

对于随机化战略的最小生成树算法，不会因为边的选择是随机的，这条边就一定属于某棵最小生成树。因此，我们在随机挑选边的时候也需要进行某种测试，以衡量此边是否合适。而为了进行此种衡量，我们需要对边进行某种划分，但这种划分和测试本身必须在线性级别上。如何做到这一点呢？鉴于篇幅限制，有兴趣的读者请参阅《算法之道：从无有到无穷》。

对于数量级$O(E+V\lg V)$和$O(E+V)$的比较，在稠密图的情况下，它们确实是一个数量级。但如果图是稀疏的或者E和V是一个数量级，则$O(E+V\lg V)$归结为$O(V\lg V)$，$O(E+V)$归结为$O(V)$。如果节点数很多，$O(V\lg V)$将显著高于$O(V)$。而且，贪婪选择战略的$O(E+V\lg V)$时间成本是在使用复杂的斐波拉契堆结构情况下，而且是摊销时间！因此，从多方面考虑，随机化战略优于贪婪选择战略。

由本文可见，对最小生成树问题的反复推敲可以串起算法里面的全部设计战略。当这些战略被串起时，我们所看到的就不仅仅是独立分散的个体战略，而是一条平时所不见的算法之道！虽然若隐若现，但对慧眼来说，它确确实实存在。这就是“形而之上谓之道”的算法境界。

作者简介 | 邹恒明

美国密歇根大学博士。曾任职于美国IBM、国家数据公司、朗讯、EMC公司。目前执教于上海交通大学，著有《算法之道》、《计算机的心智：操作系统之哲学原理》、《有备无患：信息系统之灾难应对》。

■ 责任编辑：董世晓（dongsx@csdn.net）

篱笆社区的架构变迁

■ 文 / 王一

作者详细介绍了篱笆社区从2004年创立开始，直到2010年的架构变迁，从这些变迁中，我们能深切体会到企业的技术选择与发展之路。

篱笆社区创始于2002年9月9日，是一个主要以分享装修、结婚、育儿以及相关家庭生活消费经验的交流社区。社区的帖子不物理删除，任何年代的主题都可以回复，会员习惯将自己的装修、结婚、育儿日记都写在帖子中，这也使得社区中的长帖、图片帖比较多。

架构变迁

篱笆社区采用了典型的LAMP环境，应用的模型也不复杂（如图1），大部分的架构变迁都围绕解决数据和并发快速增长导致的各种性能瓶颈而展开，从上线至今大致经历了三个阶段。

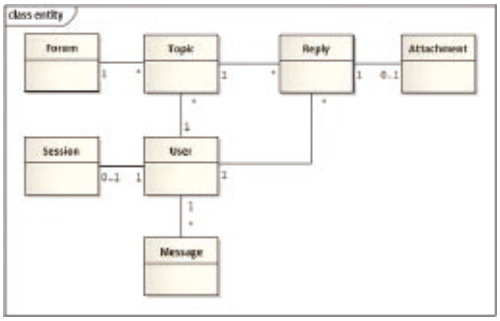


图1　篱笆社区的应用模型

起步阶段（2002-2004年）

篱笆社区起步阶段和大部分个人社区类似，是一个在租用的虚拟空间上架设开源版本的论坛。随着人气的提高，直到2003年才专门购置了一台普通宝德服务器托管在电信IDC机房，截至2004年底注册会员数量120,000人、主题数量163,000个、回帖数量3,300,000个，最高在线人数3,000人，期间系统基本上没有进行过任何的改进和优化（图2为2004年底架构示意图）。

有限条件下的架构调整（2005-2006年）

在2005-2006年这两年间，社区的各项数据

图2　2004年底架构示意图

都在快速增长，每年的会员、发帖、回帖和站内短信数量的增长率均超过300%，遇到的性能瓶颈主要集中在数据库、存储容量及带宽流量上。一些典型瓶颈问题及解决策略分析如下。

- 2005年3月，在线人数4,000+，MySQL进程堆积，Session表读写压力大。

策略：拆分Session数据库，将Session表改为MySQL的内存表，缺点是数据库服务器重启后Session数据将被清空。

- 2005年6月，在线人数6,000+，MySQL进程堆积，进程中有大量Topic表的LIKE查询，且Reply表在数据超过4,000,000后写操作时表锁严重。

策略：新增一台服务器用来作为专职的DB服务器，首先拆分出Topic和Reply数据库，然后按照Forum拆分成多张Topic和Reply表，通过此举大大缩短了表锁时间。

- 2006年3月，在线人数10,000+，数据库写操作表锁严重，DB服务器Query的峰值达到100/s。

策略：首先将社区非关键应用的数据（如站内短信、数据统计等）迁移至PC机，DB服务器硬件升级后，再利用不同的端口开启3个MySQL实例用于处理User、Topic和Reply的数据读写，在服务器性能允许情况下隔离并发的

MyISAM表锁。

- 2006年5月，在线人数12,000+，现象同3月份。

策略：将所有主题下的回帖页面生成静态HTML文件，容量大约30GB左右，在非数据库主从的情况下做到读写分离，从而降低Reply数据库的读写压力，为此社区的Web服务器做了一次全面的硬件升级来应对大量访问的I/O开销。

- 2006年8月，在线人数14,000+，附件的图片无法访问，100M带宽不够用。

策略：首先新增了一台图片文件服务器和一条单独百兆线路以应对社区附件图片的上传和访问压力，同时在Apache上使用Expires模块将图片在会员的浏览器客户端缓存30天，最后又增加了两台PC机做Squid的图片缓存。

- 2006年11月，在线人数16,000+，现象同8月份，200M带宽也不够用。

策略：2006年底是社区最为困难的时期，不仅仅因为带宽不够用，而且用于存储附件图片的服务器空间也将很快用完，加之机柜基本处于超荷的运转状态，运维和开发人员时时如履薄冰，唯有加大IT投入才能够解决这些问题。

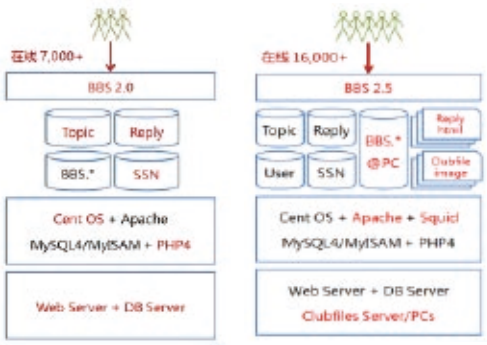


图3　2005年底（左）和2006年底（右）架构示意图

加大IT投入后架构调整（2007年至今）

2007年初公司加大IT投入，形成了以苏州科技园电信中心机房加吴江电信、苏州网通分流机房的格局，中心机房负责所有的动态请求和处理、分流机房负责图片缓存和加速，另外还采购了容量为2.8TB的HP MSA盘柜用于存储附件图片，将关键应用涉及到的User、Topic、Reply数据分布到不同服务器做主从和读写分离。随着业务的发展同样遇到了许多性能瓶颈问题，这个阶段遇到的典型问题及解决方案分析如图4所示。

- 2007年7月，在线人数22,000+，社区静态化HTML文件的弊病凸显，在苏州交易区Reply

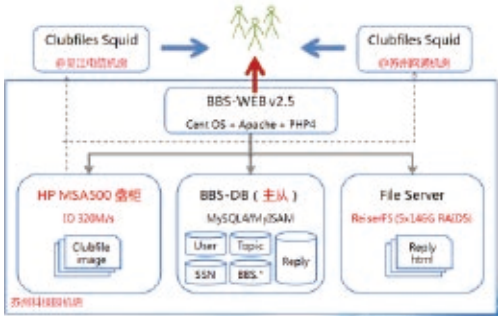


图4　2007年初架构示意图

表物理文件超过4GB时写数据延时5秒左右，导致生成的静态文件经常出错，需要管理人员干预、进行手工修复。

策略：最初尝试将Reply DB服务器升级至8GB内存，但效果并不明显、延时从最初的5秒减少到4秒，仍无法满足应用要求，只能采取粒度更小的数据拆分方式。首先想到是按照Topic拆分，当时2,000,000+的主题意味着2,000,000+的数据表，MySQL显然不合适，后来仓促之间选择了SQLite数据库、论坛回归动态模式，为配合更高的I/O读写级别要求，专门增加了刀片服务器和EVA光纤级别存储的盘柜。

- 2008年4月，在线人数30,000+，在高并发下主题超过20,000回帖的SQLite表写数据有丢失。

策略：Reply表从SQLite重新回归MySQL，部分Forum超过10,000,000记录的表按照年、月再进行拆分，缺点是需要运维和开发人员预先判断后，手工进行数据库层的读写配置修改。

- 2008年11月，需要解决按照User查询回帖记录。

策略：首先尝试使用3台MySQL5数据库服务器的集群方式，但因模拟生产环境并发下服务器I/O太高而放弃，最终使用的按照UserID Hash的1024个分区表解决。

- 2009年2月，在线人数38,000+，又有4个讨论区Reply数据超过10,000,000，6个讨论区Reply数据超过8,000,000，人工干预数据拆分对后台管理影响很大，是否有一劳永逸的解决方案?

策略：MySQL5的分区给了我们一些启发，因为MySQL5的表分区数量最大值就是1024，肯定不能满足现有性能要求，只能通过在应用层面进行处理。当时整个Reply数据大约在100GB，按照历史经验对MyISAM表10,000,000数据、4GB大小极限和数据增长速率进行估算后，我们采取按照TopicID Hash分区存在500 Db × 200 Table中。

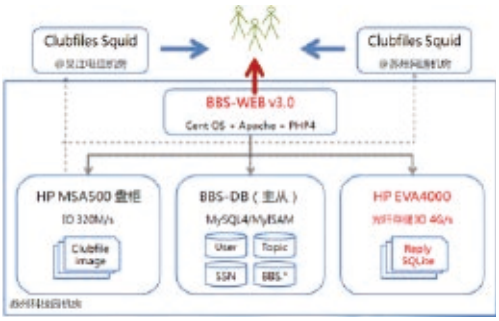


图5　2007年底架构示意图

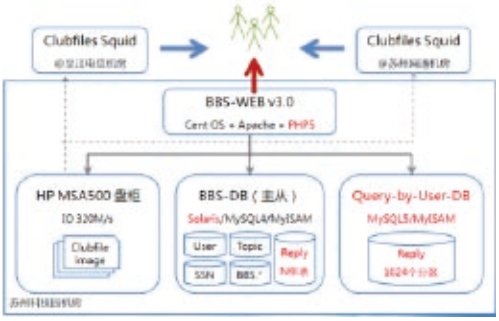


图6　2008年底架构示意图

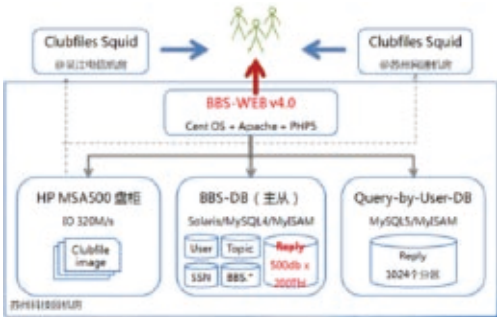


图7　2009年底架构示意图

● 2010年4月，社区站内短信表单表超过55,000,000，容量超过22GB，如何进行数据拆分提高可靠性和可用性?

策略：站内短信是典型的Key-Value应用，非常适合NoSQL。我们选择Flare作为本次调整的底层框架，Flare不仅支持Memcached协议，还包括Master-Master、Master-Slave模式，官方数据达到2,000,000,000Key和500,000Query/s。然而在使用刀片服务器测试过程中实际的数据和官方相差甚远，5台Master-Master模式数据相差4,000倍，5组Master-Slave模式数据相差200倍，可能的原因一是刀片服务器硬盘I/O性能一般，原因二是测试时使用Value的10K字节数过大，综合比较下来Flare实现成本过高。

由于站内短信应用在日常访问、数据读写的压力并不大的特点，我们使用Flare底层的存储文件Tokyo Cabinet + PHP提供RESTful接口，按照每1,000,000 Message保存到1个TC的Hash DB的方式进行应用迁移。

经验总结

之前架构调整都是围绕社区数据量和并发量最大的Reply表进行，数据库从MySQL→SQLite→MySQL，页面从动态化→静态化→动态化，数据也按照不同的应用维度拆库拆表，直到最后的100,000张表，其实这些方式都是为了避免MyISAM表10,000,000数据和4GB大小的极限值。

数据拆分这种对性能提升最直接的方式是有弊端的，最为明显的是应用方面不得不被动跟随拆分而不断调整，运维方面也为临时、不成熟的拆分方案付出过高的成本。

除拆分之外还有一些对性能提升有帮助的策略也应该优先考虑。使用缓存一方面解决了高并发下数据读写问题，另一方面还可以减小数据表中的冗余数据。以社区最近一次架构调整为例，用600MB的Memcache替代40GB冗余数据，效果十分显著。

就MySQL数据库引擎而言，InnoDB在篱笆社区数据规模下性能优于MyISAM，目前在我们攻克InnoDB下数据热备和还原方案后，2011年将进行全面的替换。

干净的代码是一切优秀架构、应用和产品的基础，在系统结构变迁的同时，感受最深的是代码质量往往是系统调整的最大障碍，一方面PHP语言本身约束较弱，导致大部分PHP代码描述性、可读性不高，另一方面在原来资源受限制的情况下往往采取了一些极端方式。

经过社区开发和架构人员一年多来使用结对编程、代码重构等敏捷方法实践后得到的结论，牺牲20%~30%代码运行效率来提高整个应用代码质量是值得的，此举在很大幅度上提高了性能瓶颈侦测与架构调整的响应时间。

作者简介 | 王一

篱笆网产品技术部Web应用架构师。有8年互联网、ERP软件开发和系统架构经验，一直关注高可扩展性的系统设计。2009年加入篱笆网担任Web应用架构师，目前负责社区一线架构调整和性能优化工作。

■ 责任编辑：董世晓（dongsx@csdn.net）

云计算环境下的应用架构设计

■ 文 / 方国伟

作者从云计算环境下应用的特点出发，分析了在云计算环境下应用程序开发设计的一些新变化。根据这些特点，本文提出一个“自我感知应用”（Self-Sensing Application）的新概念，接着以Windows Azure平台为例阐述如何实现自我感知应用。

多年来应用程序开发者和架构师们都在努力设计一种既能够在功能上满足当前业务需求，又能够适应用户需求发生变化或者能够在可预见的将来适应环境变化的应用。尤其是在互联网领域，架构师都在努力让自己设计的应用具有比较强的扩展能力，能够跟得上用户不断增长或者出现突发请求的一些情况。在传统的Web应用设计中，我们在架构上一般采用基于多层架构的设计，在Web层中大量使用了负载均衡等技术。一般我们的处理方式都是在应用程序设计好之后，在应用部署的过程中事先把环境配置好，而应用程序在运行过程配置都是不变的。但是，随着云计算时代的到来，我们面对一些新的挑战，相应的应用程序设计方式随之发生了一些变化。我们首先从云计算的技术特点开始讨论应用的变化。

从技术角度看云计算的特点

毫无疑问，云计算是目前信息产业中讨论得最多的话题。虽然大家对于云计算还没有一致定义，但是对于云计算的一些特点，相关的服务模型等内容日渐趋于统一。在讨论云计算应用架构特点之前，我们先从技术的角度来讨论一下云计算本身的一些特点。

按需服务

云计算是一个把信息技术作为服务（IT as a Service）提供的一种方式。这种服务的概念都是从消费方（用户）角度出发，而不是从服务提供方出发考虑问题，因此，云计算要求按需服务，即用户可以根据需求即时得到服务。从这个角度讲，云计算就像我们公共服务中的自来水、电、煤气一样，集中供应并按需服务和计费。

资源池

云计算的一个好处是提高资源的利用率，而这一般需要通过共享的方式来达到这个目的。这里可以类比一下我们日常吃饭中的自助餐和桌餐的差别。如果需要共享就需要先把资源集中到一个公共的资源池中。根据这个资源池中资源的类别，我们把云计算的服务模型分为三种，即所谓的SPI模型，如表1所示。

表1　SPI模型

资源类别	云计算服务模型(SPI)
应用程序	Software as a Service (SaaS)
系统平台	Platform as a Service (PaaS)
基础设施	Infrastructure as a Service (IaaS)

高可扩展性

云计算平台的资源池相对于单个用户的需求而言是比较大的，因此考虑到会有大量不同用户共用一个资源池，他们之间的资源使用模式一般存在一定的互补性，所以对于某个用户的需求而言，云计算具有很高的扩展性。另外，云计算平台在做架构设计的时候，都会考虑到如何让用户可以平滑扩展他们的资源需求，比如计算资源、存储资源等。

弹性服务

弹性服务指的是云计算的资源分配可以根据应用访问具体情况进行动态地调整。也正因

如此，云计算对于非恒定需求的应用，比如需求波动很大、阶段性需求等，具有非常好的应用效果。在云计算的环境中，资源的扩展方式可以分为两大类：一类是事先可以预测的，比如一些季节性的需求；另一类是完全基于某种规则实时动态调整的。无论是哪一种，都要求云计算平台提供弹性的服务。

自服务、自动化和虚拟化

与日常生活中的ATM等自服务类似，在云计算中自服务同样是降低服务成本、提高服务便捷性的一种途径。对于云计算服务提供方来说，自服务就要求提供尽量简单的用户操作界面，简化用户操作，降低用户使用服务的难度，只有这样自服务才能被用户所接受。而且由于是人机交互，因此服务响应速度的要求就会更高。所有这一切都需要通过后台自动化的方式才能实现，也就是说后台自动化是前台自服务的保障。因此从这个意义上来讲，自服务是目的，而自动化则是手段。虚拟化的本质是解耦，是一种把资源从硬件束缚中解放出来的方式，从而使得资源的动态分配成为可能。这几个概念的关系可以用图1表示。

图1　自服务、自动化和虚拟化之间的关系

云计算自服务一般是通过Web门户来体现，就像在亚马逊的云计算服务中，用户通过自服务门户预定需要的计算资源就像在其电子商务网站上订购一本书一样方便。自动化一般通过程序和大量自动化脚本来实现，使得前端自服务界面用户触发的操作后台平台能够自动化完成，并及时响应，从而保证良好的用户体验。

服务可度量

管理学之父德鲁克曾经说过一句名言：“如果你不能测量它，你就不能管理它”。云计算作为服务提供的方式，需要对服务进行度量。一般服务提供方和用户之间需要有一个服务水平协议（SLA）。这样对于私有云来说，可以根据服务情况进行内部费用核算。而对于公有云来说，服务可度量就是计费的前提，然后根据实际使用量来进行计费。

云计算应用的特点

从前面的描述我们可以看到云计算给应用程序带来的一些挑战，那就是应用程序如何在云计算环境下充分利用云计算平台的一些特点来更好地满足用户需求。云计算应用要能够利用云计算环境中可动态扩展的资源，构建一个具有弹性的高可用的应用程序。下面我们分别讨论一下云计算环境下应用程序的特点和要求。

自动化要求

自动化是人类的梦想，而计算机对自动化领域的发展有着巨大的影响，它极大地提高了工作和生产效率。在云计算环境下，自动化要求实际上是对计算工作本身的一个自动化改变。云计算的自动化可以赋予用户对平台基础架构的资源配置任务进行全面统筹的能力，并实现对资源的动态分配以提高管理效率、减少人为错误并加快用户对资源请求的响应速度。应用程序在设计的时候要能充分利用云计算环境的自动化特性，从而使得应用程序可以在很少或没有人工干预的情况下，自动适应需求的变化。

分布式计算

大部分云计算平台都是用廉价和标准的计算机硬件构成，然后通过云计算软件的方式在计算能力、可靠性等方面来达到传统的大型计算机的水准。也就是说在云计算环境下，资源池通常是通过分布式软硬件方式来实现。因此云计算应用程序的运行往往涉及到多个计算资源。无论是计算还是存储需求，应用程序一般都会涉及到多个节点，这样在设计的时候要考虑并行设计的思想或采用分布计算的方式。比如，有些云计算应用可以根据计算的要求，采用类似MapReduce的编程模型。

松耦合

无论是功能上还是性能上云计算对应用的灵活性提出了更高的要求。这就要求应用程序在设计的时候要考虑松耦合的架构。耦合度与灵活性一般是相反的，即耦合度越高灵活性越低，而耦合度越低灵活性越高。因此，在做云计算应用架构设计的时候，一般要追求松耦合的设计。比如，在做Web应用设计的时候，对于用户状态的保持就需要尽量采用无状态的方式来设计，这样应用程序的水平扩展能力比较好。

数据存储方式

在传统的应用设计中，我们一般采用关系型数据库来存储数据。但是在云计算环境下，尤其是对于互联网应用，存在两个需要面对的问题：一个是云计算环境下的数据量都比较大，传统的关系型数据库面临数据扩展能力的挑战；另一个是许多应用对于数据存储的要求更多体现在非结构化数据或者是半结构化数据的存储上面。因此，大多数云计算平台都会提供针对非结构化和半结构化的数据存储方式。这样应用程序的架构需要针对新的数据存储方式作出调整。

上面描述的一些应用特点对我们开发和设计应用程序会带来许多影响。一个是应用程序在设计的过程中不仅仅需要考虑操作系统平台或中间件级别的编程接口，还要针对其运行的云平台的接口来对应用程序进行设计。另外一个比较大的影响是有一些工作量从平台产品转移到了应用程序的开发设计人员。比如说，在采用半结构化数据存储的时候，开发设计人员需要处理数据的一致性问题。还有，在云计算平台上如果想要得到比较好的性能，开发设计人员往往还需要对数据的分区进行特别设计，或需要采用一些并行设计的算法等。

具有自我感知能力的应用

传统的基础架构或系统平台中的资源都不能动态配置，因此应用程序在设计的时候主要考虑自身的业务逻辑的实现。应用程序本身的监控和管理都是通过其他系统管理软件如System Center、Tivoli等来实现。有一些管理得比较好的应用程序，可以通过这些系统管理软件实现部分资源的动态调整。但是，这些传统的应用程序本身对底层平台的运行情况是没有任何感知的。随着云计算的出现，应用程序本身的自动化逐渐成为可能。我把这些具有运行环境感知功能的应用程序称之为“自我感知应用”（Self-Sensing Application）。

自我感知应用的出现是应用程序发展的一种趋势，是自动化在应用程序运行过程中的一种体现。在做传统应用程序设计的时候，我们把主要的精力都放在功能性的需求方面，但是对于一些非功能性的需求往往采用手工配置和采用第三方工具的方式来实现。云计算平台的出现使得计算平台的资源具有可编程的特性，因此我们在应用程序的架构设计中可以通过基础架构平台的一些接口来感知应用程序的实际运行情况，并可以结合访问情况对应用程序的运行资源进行动态调整，以实现完善的自动化程序运行。

云计算平台具有可编程的资源分配，因此我们还可以设定自动化的部署过程，也就是让应用程序的可以自动化地完成应用程序的部署、升级等工作。自动化部署也是应用程序动态扩展的一个前提。这样当需要更多计算实例来处理更多的用户请求的时候，新的计算实例可以动态生成出来并自动启用。

自我感知应用是应用程序朝着成为具有独立、自治单元的方式发展的一种体现。应用架构师在做设计的时候，面对的是底层抽象的、几乎无限的计算资源，而不是传统意义上的物理资源。从另外一个角度看这种设计方式也是应用程序与底层计算平台松耦合的一种体现，从而使得应用程序不绑定具体物理硬件。云计算平台从底层提供几乎无限的计算、存储和网络资源，其上的应用程序就像一个个具有人工智能的独立单元，他们在完成自身业务工作之外，还能够具有自我管理和自我修复的功能。

Windows Azure应用程序的自我感知实现

对于一个自我感知应用而言，一方面它要能够感知底层运行平台的一些环境信息，另一方面它还需要一种机制能够把自身运行要求转递给底层运行环境。Windows Azure上应用程序由代码和基于XML的配置文件两部分组成。应用程序可以通过配置文件把自身运行要求传

递给Windows Azure，确切地讲是提交给Fabric控制器。但是，应用程序如何才能感知环境的一些变化呢？加入配置文件更新之后应用程序如何才能得到通知并作出响应呢？这里就要用到Windows Azure提供的服务运行时编程接口（Service Runtime API）。

Windows Azure的服务运行时编程接口最常用的使用方式就是帮助应用程序了解应用服务和应用所在的Role实例的信息，包括：

- 它能够让应用程序访问在服务配置文件和服务定义文件中的最新服务配置信息。当配置文件的信息被更新的时候，服务运行时编程接口能够保证返回最新的配置信息。
- 它能够应用得到最新的服务拓扑结构，比如哪些Role实例在运行，每种Role类型有多少实例等。
- 由于Worker Role中的代码运行周期有点类似于有限状态自动机的处理方式，服务运行时编程接口能够帮助应用得到Worker Role实例的生命周期信息。

服务运行时编程接口可以通过两种方式使用。对于.NET托管代码，Windows Azure的SDK中包含一个名为Microsoft.WindowsAzure.ServiceRuntime.dll，当用Visual Studio新生成一个云服务项目时它会被自动引用。而对于本地代码，可以通过使用SDK中的头文件和库文件就可以用C来调用这个编程接口了。

服务运行时编程接口是应用程序本身用来得到自身及其运行环境信息用的，但是如果应用程序之外，比如说一个管理工具要得到指定应用程序的信息，那么一般需要利用另外一个称为服务管理的编程接口（Service Management API）。这两个编程接口在功能有重合的地方，它们之间的最大区别在于服务运行时编程接口在Windows Azure中运行，而服务管理的编程接口一般在Windows Azure之外运行，它更多的用在那些针对Windows Azure的管理工具开发当中。

结束语

云计算无疑已受到极大的关注，但是它还是一个相对较新的概念，相关的技术正处于快速发展的过程中。我们已经可以看到云计算对IT行业的硬件模型、应用模型和用户体验等方面带来了革命性的影响。从应用模型的角度来看，云计算平台的出现使得开发人员可以快速构建高可用的、可以几乎无限扩展的应用。随着云计算相关技术的进一步发展，它将使我们能够进一步简化开发自我感知应用的工作，另外非功能性的一些要求也将更多通过配置文件而不是代码来实现。

人们对于信息技术需求的发展始终没有改变，那就是追求可靠、便捷、易用的信息应用。当我们把一个冰箱连接到电源插座的时候，我们从来不会去考虑这个电力是从哪个发电站来的，我们也不需要成为一个电力专家来使用这个冰箱。云计算的出现让信息技术往这个方向更进了一步。我们认为应用程序将逐渐演变成具有自我感知能力的应用，成为一个能够根据平台环境和用户请求进行自我调整和自我修复的自治单元。也许不远的将来，整个互联网就像一台巨大的计算机，其上提供无限的计算资源和服务，人们使用其上的应用程序就像我们现在使用自来水、电力那样方便、自然。

参考文献

[1] David Chappell, Introducing Windows Azure, http://www.davidchappell.com/writing/white_papers/Introducing_Windows_Azure_v1.2--Chappell.pdf

[2] Peter Mell, Tim Grance, The NIST Definition of Cloud Computing, http://csrc.nist.gov/groups/SNS/cloud-computing/cloud-def-v15.doc

[3] Alexander Lenk, Markus Klems, Jens Nimis, Stefan Tai, Thomas Sandholm, What’s Inside the Cloud? An Architectural Map of the Cloud Landscape, CLOUD’09, May 23, 2009, Vancouver, Canada

作者简介 | 方国伟

软件架构资深顾问，目前在微软开发工具及平台事业部负责微软云计算解决方案的技术推广工作。曾先后担任IBM（中国）有限公司软件部资深软件工程师、微软企业和合作伙伴部制造业客户技术经理。

■ 责任编辑：董世晓（dongsx@csdn.net）

深入探析 Android Market 大势

■ 文 / 刘铁锋

Android Market推出已两年有余，其发展看起来却不瘟不火。原因何在？怎样解决？作者根据对多年来移动互联网的分析与判断，分享了自己的见解。

从2008年底Android Market发布以来，业界就开始拿它和App Store进行比较。App Store发展中，不断创造出来的百万富翁的传奇，让大家艳羡不已。无数开发者前赴后继地进入这个领域淘金。而Android Market的发展却看起来不瘟不火，似乎没有太多令人激动的事件发生。即便如此，Android Market的程序发展速度还是相当迅猛。图1展示了Android Market从2009年7月~2011年1月的每月程序发布情况。

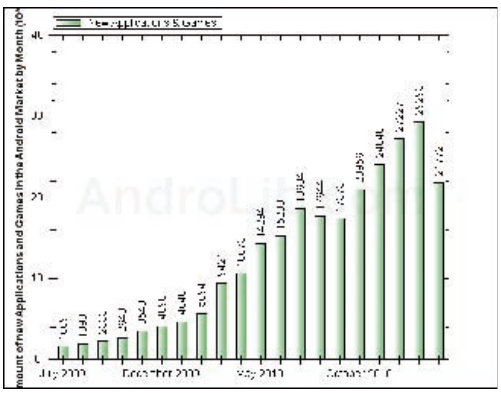


图1　Android Market程序发布数量

从图1可以看出，在不到两年的时间里，Android Market上的程序发布数量有了显著增长。从每月不到2000、平均每天不到70款程序，发展到每月29000多、平均每天近千款程序，也就是说平均每两分钟就会有1款程序发布。相比于App Store三年发布30多万款程序，Android Market在两年多的时间里发布了20万款程序，这个数量不可谓不多。要知道，Android Market刚上线时只有区区34款应用和9款游戏，到2009年3月，也不过才有2300款程序。截止2009年底，Android Market有2万款程序。而到了2010年，Android Market发展迅猛，单单在2010年的最后三个月，就增加了8万款程序。

同时，Android设备也得到了跨越式发展，市面上已有上百款造型各异的手机。摩托罗拉已经借助Android手机扭亏为盈，也有消息表明HTC在2011年期望的Android发货量将达到2400万台，并且其市值也在近期超过摩托罗拉，可以说Android绝对功不可没。此外，平板电脑也是相当大的一块市场，各大厂商已然开始发力。

毫无疑问，Android的发展形势一片大好。

Android Market的程序下载情况

Android终端的市场表现已经非常出色了，大有赶超iPhone之势。那么整个产业中的非常重要的一环——Android Market，这个作为发行和催生出创意应用的渠道的发展情况又如何呢？

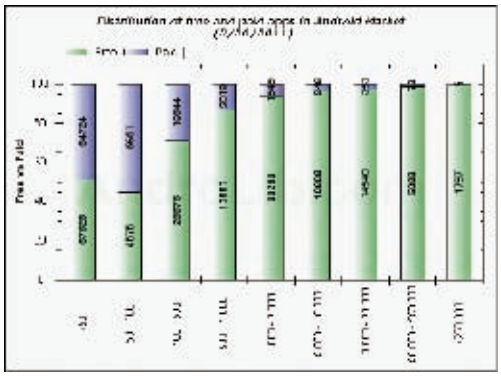


图2　Android Market的销售情况（截止到2011年2月20日）

从图2可以看出，这两年来Android Market付费程序的销售情况的确不太能拿上台面。仅有五款程序销售量超过了25万，且销售量在5万~25万的程序也不过38个。想在这里看到类似App Store上的传奇式的付费量，似乎很难让人信服。

如果说付费程序的情况不够好，那么免费程序又如何呢？同样也只有近6000种程序的下载量超过了5万。考虑到这两年来Android终端的迅猛增加，程序的下载情况的确很值得玩味。

图3是一个更加详细的程序下载量分布图。

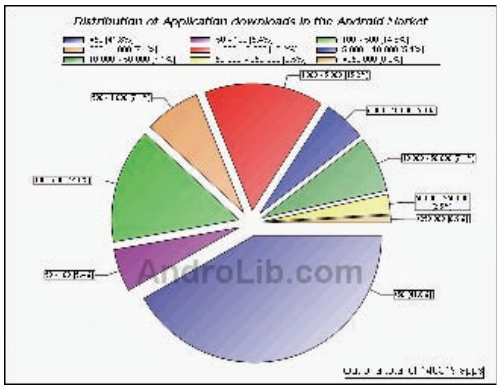


图3 更加详细的程序下载量分布图

从图3可以发现，有近半数的程序下载数量小于50，基本上就是无人问津。而程序下载量在5万以上的，只占所有程序的3.4%；下载量在1万以上的，也只有7.1%。相比于2000万~3000万的Android终端数量，这个数据可以说比较惨淡。

直观地来看这些下载的数据，可能会得出似乎Android用户不太喜欢下载程序的结论。那么，这些程序的评分情况如何？如图4所示。

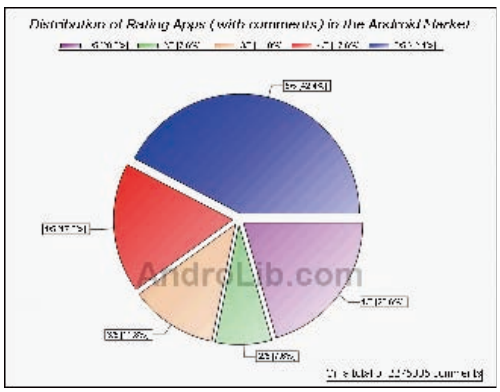


图4 Android Market程序的评分情况

可以看出，低于4分（事实上低于4分代表用户对这个程序不太满意）的比例达到40%，而评5分的则高达42.4%。这是否意味着的确有这么多用户对目前的程序如此满意呢？对于超过50%的下载量为50以下的程序，是否可以推测出这些5分的分数更多地来自于开发者，而不是用户呢？

通过对以上数据的分析，不可避免地会发现，Android Market的实际发展情况并非像设想的那样美好，用户对程序的接受程度依然有限。

Android Market的现状

从前面的数据中可以看到，Android Market上靠销售的盈利能力是一个不小的问题。作为一个开发者在开始审视Android Market的时候，有以下应该需要考虑的因素。

- 是否能够让开发者能够赚到钱，从而催生出更多有价值的应用。
- 是否有足够多创新的东西能够吸引用户一而再、再而三地有购买和挑选的欲望。
- 对于新开发者来说，能否有更好的方式让其创意的应用脱颖而出。

结合以上的几个因素，以及目前Android Market的发展现状，我们来分析一下Android Market所面临的问题以及挑战。

程序销售的问题

Android Market程序销售不佳原因如下。

支付的问题

这是导致Android Market销售不佳的最主要的原因之一。Google的Check Out的支付及使用方式不仅需要用户做相当多复杂的操作，而且与App Store绑定信用卡的支付方式相比，其付款体验不太理想。此外，Google Checkout所支持的国家和地区也是一个限制因素。

24小时退款策略

Android Market的程序绝大部分都提供了功能完整的免费版本，而24小时退货策略也使Android Market的退货率居高不下，直接导致开发者很难通过这个方式真正有所收益。当然，最近也看到Google在退款策略上做了不少改进。

盗版问题

根据我的经验，好程序的盗版版本，几乎在24小时之内就会在互联网上泛滥。不仅如此，国外同样也有为数众多的网站，提供所谓的$99包年服务：只要$99，就可以下载Android Market的所有付费程序，这让开发者情何以堪？因此，如果用户养成了免费的习惯，再加上开发者之间的竞争，这个问题将会始终困扰Android开发者。

广告的问题

在Google的App Market的游戏规则里，广告毫无疑问是留给开发者的另外一条活路，也是Google 更加擅长的一条路。在Apple的iAds推出时，有一个做手电筒应用的开发者一天的广告收入竟然多达上千美元。相比来说，Google Adsense同样拥有来自于Angry Bird的好消息，据称它每月的广告收入会达到百万美元。当然，这些收益是建立在其数千万的用户基础之上的。

如果你的程序数量不够大，广告的确可以让你每天都有收益，但收益是否足够大，这是每个开发者自己要去衡量的问题。

App Ranking 问题

仔细观察Android Market的排行榜，就会发现排行榜上的程序似乎从来没变过。经过分析，Android Market在程序排名上有如下问题。

新程序胜出方式单一

在Android Market上，目前有如下排名方式：Featured、Hottest、Just In。Featured是一个很好的推广途径，但数量不多，导致用户无法有效地进行选择。Just In分类作为一个新程序上榜和推荐的机会，已经被众多的程序数量给抹杀了。平均每小时60种程序，一小时之后你的程序就消失在了茫茫的程序大海中，再也不会出现。

传统厂商的聚集效应

Android Market上排名靠前的程序被大公司所占据，而很难看到小的、有创意的程序出现。反观App Store：很少看到如此多的大品牌出现，取而代之的是各种小的创意性产品。相比之下，小的开发者是否能够在Android Market淘到金、如何面对大公司的压力，这也会是一个问题。

设备以及版本的分裂

如果用户给你评上一个1分，并且反馈是“在我的xx设备上跑不起来”，而这个xx设备却是一个发货量极少或者是用户自行刷过ROM的机型，你心中的郁闷确实会难以言表。这就是目前Android终端面临的最大问题之一：分辨率、操作系统，甚至厂商定制的第三方UI。开发、适配逐渐成为开发者的噩梦。此外，各种眼花缭乱的第三方Market和发行渠道对用户又有多大帮助呢?

Android Market的发展挑战

通过前文的分析，不难理解和发现Android Market的现状。尽管表面上形势一片良好，然而对开发者来说，却是冷暖自知。在这里，我斗胆提出Android Market亟待解决的问题。

付费的问题

从本质上来说，尽管Google的确会付出更多的努力来解决付费的问题，但是在有众多试用的免费版本的情况下，付费路线是否能够培养起来，的确需要实践来检验。Google试图希望将众多高质量的iPhone用户吸引到Android平台上来，Angry Bird作为最早的一批已经迁移过来，短短几天下载量就超过200万。这充分说明，Android用户并非不愿意下载程序。Angry Bird免费150关的架势，似乎就根本没打算收费，其CEO似乎也在博客上言明不会在Android收费。但是，并不是每个开发者都会这么走运。

Market的程序筛选机制

Android Market的繁荣和发展，最终还是得依靠千千万万的开发者。如何让好的创意能够有更多的渠道表现出来，而不是被各大公司的程序一成不变地垄断排行榜的前列，这的确是Android Market发展的最大挑战之一，也是直接影响到开发者信心的渠道之一。

版本分类以及筛选问题

让一个规模很小的开发者去适配各种性能不一的机型，的确会是不小的挑战。目前Android Market采用的是只针对屏幕大小的筛选方法，还不注意帮助开发者更好地筛选出不同的机型。改进这个问题，也会是一个帮助到开发者的方向。

展望

尽管Android Market看起来问题重重，但机遇与挑战并存。相关研究报告表明，Android手机数量在2011年将会增长683%，设备数量甚至会增加到9千多万台。再加上平板电脑的爆发，机会实在不容小视。

同时，面对这万亿的移动互联网市场，以及新的产业变革的机会，Android Market上的竞争激烈程度会比开发者想象得更加残酷。这短短的一年多时间，已经见证了很多曾经轰动一时、在排行榜高高在上的程序的兴衰。Android Market终归会成为实力派厮杀的战场，仅靠单枪匹马、好的创意已难以取得成功。只有持续的改进和努力才可能获得用户的认可。

作者简介 | 刘铁锋

百纳信息技术有限公司CTO，目前从事移动设备软件相关开发工作，关注Android、iPhone等相关产品开发技术以及移动互联网应用。

■ 责任编辑：董世晓（dongsx@csdn.net）

在调试器中看Win7打电话回家（上）

主持人：张银奎

《软件调试》一书作者，从事软件开发和研究10余年，对IA-32架构、操作系统内核、虚拟技术，尤其对软件调试有较深入的研究。翻译（合译）作品包括《数据挖掘原理》、《机器学习》、《人工智能：复杂问题求解的结构和策略》、《观止——微软创建NT和未来的夺命狂奔》等。

过完春节，很多人又要告别家乡到外地学习或者工作了。相对于长时间的分别，偶尔的团聚显得越发珍贵和短暂。一首《常回家看看》唱出了普天下父母和游子的心声。但当今时代紧张而且快节奏的生活方式让很多人都难以实现这个美好愿望。好在还可以经常打电话回家，报声平安，叙叙家常。

从某种程度来讲，软件就像是程序员的孩子。设计阶段，苦思冥想，规划这个“孩子”长什么样、能干什么；开发阶段，全力以赴，恨不得给“孩子”打造三头六臂，希望他聪明伶俐、勇武健壮，能适应各种复杂的运行环境；测试阶段，殚精竭虑，努力发现和弥补“孩子”身上的不足。把软件发布出去后，牵肠挂肚，很想知道“孩子”在用户那里、“工作”得怎样。从事软件开发越久，这种感受就越真切。有些年轻的程序员没有建立起这种感受，写起代码来，心不在焉，很难写出好的代码。话说回来，如何知道发布出去的软件在用户那里运行情况怎样呢？答案是让它常打电话回家。如何打电话呢？通过互联网打IP电话啊！具体怎样实现呢？在接下来的两期文章中，我们就来看一下Windows 7（简称Win7）是如何实现这个目标，经常“打电话回家”的。

哪些情况打电话

在Win7的“隐私声明补充（Windows 7 Privacy Supplement）”中，详细定义了Win7“打电话回家”的44种情况，我们把每一种情况的标题列在表1中。

浏览一下便知，打电话的情况还是很多的，纵向来看，从系统安装、激活，到设备管理和应用程序兼容；横向来看，从打印传真，输入法，到时间服务、远程访问等。总之，这张表几乎囊括了用户使用计算机的所有方面。

表1　Win7“打电话回家”的44种情况

激活	即插即用扩展
审核	程序兼容性助手
BitLocker (TM) 驱动器加密	程序属性的“兼容性”选项卡
设备信息检索	属性
设备管理器	远程访问连接
动态更新	RemoteApp 和桌面连接
轻松访问中心	远程桌面连接
事件查看器	权限管理服务 (RMS) 客户端
传真	Teredo 技术
小工具	受信任的平台模块 (TPM) 服务
“游戏”文件夹	更新根证书
手写识别（仅适用于 Tablet PC）	UPnP 技术
家庭组	Windows Anytime Upgrade
输入法编辑器 (IME)	Windows 客户体验改善计划 (CEIP)
安装改进计划	Windows Defender
Internet 打印	Windows 文件关联
位置和其他传感器	Windows 帮助
Microsoft 错误报告服务	Windows ReadyBoost
网络感知	Windows 远程协助
订购照片	Windows 语音识别
家长控制	Windows 时间服务
即插即用	Windows 疑难解答

对于每一种情况，隐私声明中都描述了为什么在这种情况下要“打电话”，以及“电话”的内容是什么，具体包含如下几项内容。

- 此功能的用途：简要介绍。
- 收集、处理或传输的信息：通过“电话”，发送回去的内容。
- 信息的使用：电话内容的用途。
- 选项与控制：如何允许和禁止这类“电话”，但有些“电话”是不可以禁止的，比如激活。

大家可以通过下面的链接阅读“隐私声明

补充”条款，了解更多信息。

英文版本

http://windows.microsoft.com/en-US/windows7/windows-7-privacy-statement

中文版本

http://windows.microsoft.com/zh-CN/windows7/windows-7-privacy-statement?T1=supplement

也可以通过如下操作：打开Windows的问题报告设置面板Control Panel\System and Security\Action Center\Problem Reporting Settings，然后点击其中的Read the Windows Error Reporting privacy statement online打开“隐私声明补充”网页。

为什么要在隐私声明中包含这些内容呢？这主要是因为法律方面的要求，使“电话回家”成为用户协议的一部分，“合理合法”化。对于开发软件的程序员来说，软件好像是自己的“孩子”。但对于使用软件的用户来说，软件是他们请来的“管家和佣人”，是要代表用户的利益为用户服务的。在用户家里向自己的老家打电话和发送信息是要慎重的，尤其是话题与用户家里的情况有关的时候，即使不明确征得用户同意，也要事先做好准备，师出有名，如果用户询问起缘由来要有理有据。

何时打电话

了解了Win7都会打哪些电话后，接下来看一下它都在什么时候打电话。尽管相对其他普通软件来说，Win7是一个非常有权有势的“管家”，但是也不能在主人面前太过张扬，比如在主人很忙的时候打电话就不太好。另外，打电话的过程也不要太高调。本着这样的原则，Win7通常都是把打电话的动作安排在系统很空闲、最好是主人不在使用电脑的时候，而且打电话的动作都是在后台悄悄进行，再也不像XP时代那样显示发送过程对话框（图1）。

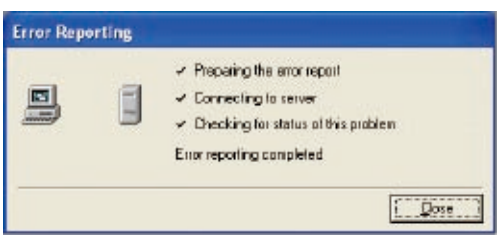


图1 XP时代的“打电话”界面

怎样做到等系统空闲时再打电话的呢？简单来说是使用系统的任务调度（Task Scheduler）机制，这里的任务调度不是指底层的线程调度，而是指宏观上的事务调度，到某一时间或者满足某一条件时自动做某件事情。在开始菜单中输入task然后选择Task Scheduler调出任务调度控制程序（图2），在左侧的树形控件中选择WPD条目，就可以看到用于自动打CEIP电话的预定任务。

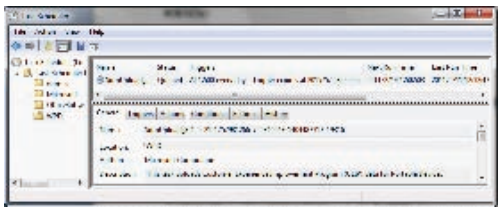


图2 自动打CEIP“电话”的预定任务

这里的WPD是Windows Portable Devices的缩写，意即便携设备，CEIP是Customer Experience Improvement Program的缩写，意即用户体验改进计划。简单来说，打这个“电话”的根本目标是为了改进便携设备的用户体验，属于表1中第28项的一部分。

用户是上帝，改善用户体验太重要了，因此从图2中可以看到，这个电话任务是每天（every day）执行一次，一直执行到2015年。任务的执行时间是12:00，注意不是夜里12点，而是中午12点，因为这个时间用户通常去吃饭了，系统有空，而且在互联网上，不会像夜里12点时很多用户会把机器关掉。

选择右侧下方面板中的Conditions页面（图3），可以看到运行这个任务的详细条件。

系统至少空闲了10分钟，确保主人不是那种忘记吃午饭的工作狂。

系统是由交流电源（AC），而不是电池（DC）供电的，防止打电话浪费宝贵的电池能量，令主人不满。

必须有网络连接，以免电话打不通。

只有以上三个条件都满足时，才会启动打电话任务，不然的话就延迟计划，等待时机成熟。

从图3中可以看到，这个预定任务的名称是SqmUpload_XXX。SQM是Software Quality Monitor的缩写，意即软件质量监控，是

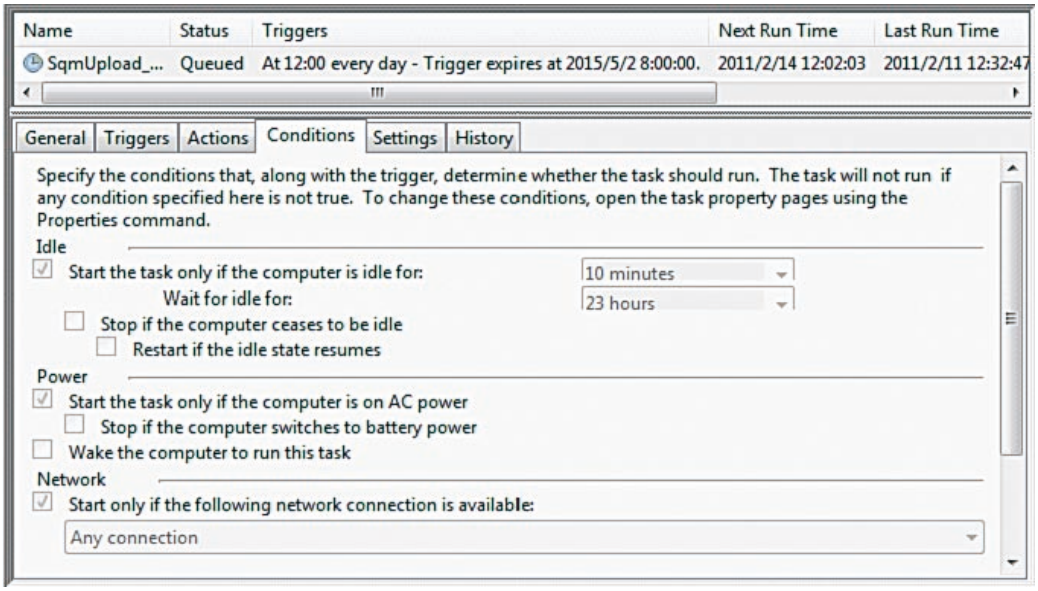


图3 启动CEIP “电话”任务的条件

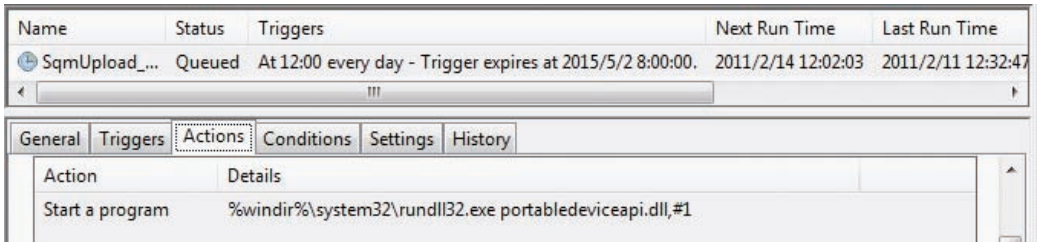


图4 启动CEIP“电话”任务的动作

Windows Vista引入的一套用于自动收集和发送软件运行信息的基础设施，我们在后面还会介绍。

如何打电话

点击图2和图3中的Actions标签切换到Actions页面（图4）便可以看到满足条件后启动这个任务时要采取的动作。

从图4可以看到，“电话”任务对应的动作是启动一个程序，这个程序的可执行文件是rundll32.exe，命令行参数是“portabledeviceapi.dll,#1”。这意味着，以rundll32.exe为载体运行动态链接库portabledeviceapi.dll中的1号（#1）函数。

跟踪打电话动作

如何跟踪上面看到的“打电话”动作呢?

第一种方法是先在注册表的Image File Execution Options表键下埋伏好调试器（图5，参见《软件调试》10.3.4节），然后守株待兔，等任务调度器时机成熟时启动rundll32.exe进程，触发注册表里的埋伏，自动启动到调试器。但是这样做可能需要等待很长时间。

第二种方法是手工启动调试器，然后打开Rundll32.exe，并指定“portabledeviceapi.dll,#1”作为命令行参数，然后开始调试。

第三种方法也是先埋伏好调试器（图5），然后在SqmUpload_XXX任务上点击鼠标右键，选择Run（图6），手工启动这个任务，rundll32.exe被启动，并且自动启动调试器。

以上这三种方法没有大的区别，我们就以第三种为例。切换到WinDBG窗口，应该看到rundll32.exe因为初始断点而中断到调试器中，此时portabledeviceapi.dll还没有加载。在WinDBG中执行以下命令：

```
sxe ld:portabledeviceapi
```

目的是当portabledeviceapi.dll被加载时，让进程中断下来。执行g命令，恢复进程执行，但

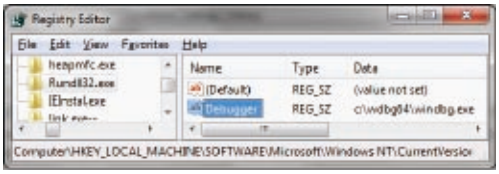


图5 通过注册表“埋伏”调试器

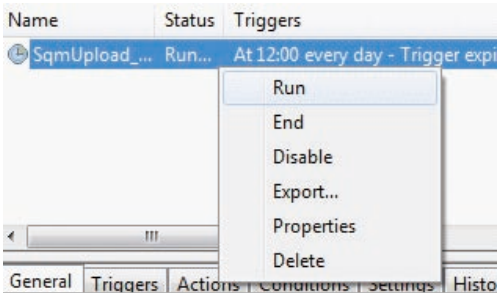


图6 手工启动“电话”任务

立刻会因为加载portabledeviceapi模块而中断下来。设置好调试符号（.symfix c:\symbols）后，“执行x portabledeviceapi!*upload*”命令显示这个模块中所有包含upload字样的符号，可以看到以下这些函数：

```
DEMANDLIBFUNC_SqmStartUpload
PORTABLEDEVICEAPI!WpdSqm::PerformUpload
PORTABLEDEVICEAPI!WpdPerformSqmUploadW
PORTABLEDEVICEAPI!DEMANDLIBVAR_
SqmStartUpload
```

其中的 WpdSqm::PerformUpload 就是导演“打电话”过程的主要方法，对其设置断点，然后执行 g 恢复进程运行，很快命中中断，执行 k 命令观察栈回溯，可以看到类似图 7 的函数调用关系。

```
Call Site
portabledeviceapi!WpdSqm::PerformUpload
portabledeviceapi!WpdPerformSqmUploadW+0x29
rundll32!wWinMain+0x212
```

图7 开始“打电话”的过程

执行 pc 命令，运行到下一个函数调用，可以看到 PerformUpload 调用 SHGetFolderPathAndSubDirW API 获取 Microsoft\Portable Devices 文件夹的具体位置，得到的结果通常是如下这样：

```
“C:\Users\XXX\AppData\Local\Microsoft\
Portable Devices”
```

其中的 XXX 代表当前的用户名。

再执行 pc 跟踪到下一个函数调用，是调用 PathCchAppend 函数将字符串“wpdlog*.sqm”附加到上面的路径后面，形成一个这样的字符串：

```
“C:\Users\XXX\AppData\Local\Microsoft\
Portable Devices\wpdlog*.sqm”
```

接下来的动作是动态加载 SqmApi.dll，定位其中的部分导出函数，然后执行 SqmStartUpload 函数，栈回溯如图 8 所示。

```
Call Site
SqmApi!SqmStartUpload+0xa
portabledeviceapi!DEMANDLIBFUNC_SqmStartUpload+0x59
portabledeviceapi!WpdSqm::PerformUpload+0xc3
portabledeviceapi!WpdPerformSqmUploadW+0x29
rundll32!wWinMain+0x212
```

图8 调用负责上传数据的SQM函数

SqmStartUpload 中会先加载 WinHttp.dll，构造一个 CSqmHttpUploadManager 类的实例，然后调用 CSqmHttpUploadManager 类的 AddFilesToUploadQueue 方法。

AddFilesToUploadQueue 先通过 SensApi 模块中的 IsNetworkAlive 方法判断网络是否畅通，然后通过文件搜索 API（PathFindFileName、FindFirstFile、FindNextFile）搜索刚才准备好的文件路径，每找到一个文件，便向上传队列中增加一项，每项中包含要上传文件的全路径，以及服务器的地址，默认为：

http://sqm.microsoft.com/sqm/windows/sqmserver.dll

如果启用了所谓的企业 SQM（Corporate SQM），则会从注册表中获取企业网内的一个服务器地址。

准备好了上传队列后，主线程（0 号）会调用 CreateThread 创建一个专门用于传送数据的工作线程。工作线程的入口为 SqmApi!QueueProc，对其设置一个断点后，恢复目标执行。此断点命中后，会发现当前线程不再是主线程，而是 3 号或者其他。

在工作线程中 QueueProc 会调用 SqmApi 模块中的 CSqmHttpUploadManager 类的 ProcessNextFileInQueue 方法来依次处理刚才放入队列中的项目。对于每一项，调用 _CacheSqm-Data 将 SQM 文件中的数据读入内存中，然后将 SQM 文件删除。而后 _Cache-SqmData 再调用 _UploadSqmData 方法，_UploadSqmData 中会做一些检查，比如调用 IsSizeTooBigForServer 来确保不要把太大的数据“传回家”。

检查没有问题后，_UploadSqmData 会调用 CHttpRequest 类的 UploadSQMDataAndReceiveResponse 真正与“家里通话”。至于通话的细节如何，因篇幅关系，我们下期再谈。

■ 责任编辑：董世晓（dongsx@csdn.net）

本期问题：

上期答案是：上期的答案是重构后，调试器使用调试端口与被调试进程对话，当有用户态调试事件发生时，内核会通过DebugPort字段知道此进程有用户态调试器，因此会将调试事件留给用户态调试器，不会发给内核调试器，因此也就不会导致冲突了。

本期的问题是：请大家在自己的Windows7系统上做个实验，想办法找到本文提到的wpdlog*.sqm文件，看其开头四个字节的内容是什么？

编者说明：

- 投稿信箱：contest@csdn.net
- 联系方式写在一个单独的TXT文件里，包括以下几项：
  1）姓名
  2）工作单位或学校
  3）电话联系方式
  4）邮寄地址
  5）E-mail地址
- 解答提交时间，最好早于当月15日。

敏捷教练的工具箱

■ 文 / 滕振宇

学习并不是简简单单的阅读和浏览，而是一个积累的过程，一个通过持续的学习，对自己的知识体系不断丰富、索引的过程。接下来我会从四个方面入手分享我的经验。

高质量的信息源和高效的学习

Google是一个很好的工具，通过它，我们可以找到很多很好的资源，但前提是必须先知道要搜索的关键字，没有关键字，就不知道该查什么。多数情况下，人们都是在不可能知道自己不知道什么（Unknown unknown）的状态，也就是不知道该用什么关键字去查询，因此也不会知道该去学习些什么。所有基于Google检索的模型是一种基于“拉动”的模型，而基于拉动模型的方式不可能让我自己去接触那些自己不知道的信息。

为了让自己接触到充足的信息，我需要建立一个信息网络和信息渠道，让那些可能感兴趣的信息不断“推送”到面前。推送渠道主要是以Twitter和Google Reader为基础。通过订阅感兴趣的内容源，我可以时刻了解业界正在发生的热点讨论、研究、工具、书籍，以便及时调整学习方向。

同时我所订阅的资源也起到过滤有价值信息的功能。从我欣赏和信任的专家那边推过来的信息或者被多次提到的关键字对我来说都是有价值的信息。思维导图是一种十分有效的快速学习的工具。在看书或者学到有价值的知识时，我经常会把学到的内容做成思维导图。这样我既可以对所学到的内容有一个概要的、抽象的了解，同时又可以随 时深入到细节。

思维导图桌面应用工具，我常用的是XMind。XMind可以方便地支持思维导图、鱼骨图、组织结构、树状图等多种形式，同时它提供了一些初始模板，包括头脑风暴、会议、项目管理、读书笔记、流程图等。

在线的思维导图工具，我使用Minemeister。它能够做到修改通知，甚至播放整个修改的过程，并且支持多个人同时在线头脑风暴。我经常在头脑风暴电话会议中使用这个工具。

方便的笔记管理

学习是一个构建个人知识库并且不断建立索引的过程。每次接触到有价值的知识，我都会做笔记。好的笔记应用满足以下几个需求：

- 记录多种形式的内容，包括文本、图像等。
- 方便的标签功能，这样我可以方便地定位特定专题的内容。
- 支持自由文本检索，这是标签功能的有效补充。类似于Google那样，通过一些关键字，去检索知识库。
- 同步与备份，方便地在不同的电脑以及设备之间同步。

最初我使用微软的OneNote来做笔记。早期版本的OneNote能够记录多种形式的内容，并且可以方便地用关键字查询。它的分页功能在一定程度上起到标签的功能，但是一个笔记只能放在一个标签下，除非复制多份，而多数情况下，一个笔记是需要放到几个标签下面的。由于微软的应用是以客户端为主，所有笔记都放在客户端，因此我需要自己备份这些文件。每次重新安装机器，都需要自己手动复制文件。

后来我选择了能够满足我对笔记管理工具的全部要求的Evernote。

首先，Evernote是一种客户端—服务器架构，可以使用多种客户端应用来记录笔记，而笔记会自动同步到服务器。这样的话，每次更换机器或者更新系统，我要做的只是安装Evernote客户端登录，就可以自动下载同步所有笔记。

Evernote做到了“写入既存档”，它会自动把最新的内容同步到每一个设备，这样我再也不用担心笔记丢掉。我比较反感很多应用中的“保存”功能，对于一个真正为了客户设计的系统来说，它不需要客户考虑是否“保存”这种琐事，每次客户输入，都应该自动保存在电脑中，而且客户根本不需要关心存放在什么地方，只要想找的时候能够方便地找到就可以了。

Evernote的Tag和自由文本检索也十分强大，它的自由文本检索甚至可以认出 Evernote笔记中图片里面的字符，如果图片中的文字符合检索要求，它也会可以帮我找出来。

另外Evernote支持多种包括PC、苹果电脑、手机、iPad等多种设备。当然，作为一个有心人，一定要随时记录学到的知识，随身一定要带笔和纸。

知识库的备份与版本控制

我的个人知识库不光包括日常笔记，还包括各种各样的文档，比如论文、PPT、视频、培训及咨询资料、个人资料等。我对于资料知识库的需求如下：

- 版本控制，我需要总是在最新版本的文档上做修改，但不希望用时间作为文件名。
- 保留历史，希望看到每份文档的历史。
- 占用磁盘空间不要太大。
- 资料同步。

我最初使用的工具是Subversion和TortoiseSvn，Subversion是服务器，TortoiseSvn是客户端。每次对文件做完更新，在相应目录或者文件上用鼠标右键点击，选择更新，TortoiseSvn就会把修改提交到Subversion服务器。使用 TortoiseSvn可以很方便查询提交的整个历史。与Evernote不同的是每次都需要手动提交。相对于笔记在Evernote中的自动更新功能，我更希望手动更新资料，这样就可利用提交时填写的注释，记录整个文档变化的历史。

但基于Subversion的解决方案也有问题，由于Subversion是基于服务器－客户端架构的解决方案，每次提交必须连接到Subversion服务器，而这个服务器往往是装在另外一台机器上，因此如果出差中，就没有办法提交更新。

与Subversion相比，Git是一种完全不同的版本管理工具。作为分布式版本管理工具，它的服务器总是在本机，因此我随时都可以提交文档的更新。Git的数据库比Subversion数据库要小几十倍。在Windows平台上需要安装MSysGit来安装Git，在 Mac平台上直接安装Git就可以。另一个重要原因是我希望通过使用Git来学习这种新的工具，以及了解分布式版本管理系统的工作方式。基于Git的解决方案给我带来一个新的问题——我的资料数据库和我的资料数据都存放在同一台机器上，万一机器崩溃或者丢失，就会带来不可弥补的损失。因此需要把数据库同步到其他地方。

我使用的是Mac自带的Time Machine，它会随时备份。Dropbox也是一个不错的备份方案，可以把资料同步到网上，但是对我不适合，因为我的个人资料库远远超出了Dropbox免费账户所提供的空间大小。而且Dropbox是采用自动更新的方式，这样我就很难利用手动提交时填写的注释来记录文档更新的历史。

提高工作与学习的效率

我十分反感低效的工作和学习，因此在我的常用工具箱中有不少提高电脑使用效率的工具，主要包括：时间管理、检索、快捷键、黏贴板等。

时间管理

对于时间管理来说，基本需求是保证能够专注于最重要的事情。因此我需要一个待办事项列表工具和一个能够帮助保证时间箱的工具。我使用“番茄工作法”管理自己的工作和学习时间。具体做法就把时间分成25分钟的时间箱（番茄时段），每个时间箱中排除干扰专注于做最重要的那件事情。我使用Pomodario作为时间箱工具（从下图可以看出，我写到这个地方已经花了十个25分钟）。

我用的待办事项列表工具是Remember The Milk（RTM）。它是一个在线应用，也可离线，具有丰富的标签功能，可与日历程序同步。美中不足的是不支持任务嵌套，我暂时通过标签来作区分。每次开始一个番茄时段之前，都会从RTM里面选择最重要的任务，然后在接下来的时段中，不受干扰地去完成这个任务。

检索

多数电脑安装的操作系统都是基于图形界面的，为了寻找文档往往需要打开资源管理器（在Mac上是Finder），然后资源管理器会树形展开所存储的资源。为了找到资源，需要用鼠标不断展开/缩放树节点，去寻找文件。这种做法十分低效，找一个文件往往需要半分钟。因此，迫切需要一个工具能够从电脑中快速定位并且启动文件或程序，只要简单输入关键字，就可以从电脑中把相关的资源列出来。

在Windows系统上，我会用Everything和Google桌面。Everything会将系统里所有文件索引，输入关键字就可以把文件名符合关键字的文件列出来，Everything还支持基于正则表达式和通配符的检索，资源消耗很少，速度很快。

Google桌面也可以通过关键字检索来定位文件，而且比Everything更强的是它不仅仅对文件名作匹配，也会去匹配文件中的内容。问题是Google桌面功能太多，对我来说太过于重量级，所以我以Everything为主，以Google桌面为辅。 Mac系统自带了Spotlight，它实现了同样的功能。只要按下Command + Space，就可以直接启动Spotlight输入关键字。

快捷键

使用键盘比使用鼠标效率要高得多，对于常用的操作或者命令，我都会写一些脚本，定义一些快捷键。在Windows系统上，我用的是AutoHotkey。在启动机器后，我需要启动Git，然后到GitHub上拉最新代码，本地运行脚本去编译和测试，然后启动Visual Studio打开项目。通过脚本和定义命令，我只需要按几个键就可以完成整个操作。

增强黏贴板

有一个功能实现起来不难，而且很实用，不幸的是它被Windows和Mac系统所遗忘。这两种操作系统的黏贴板仅仅能存放一个东西（文本、图像等）。如果按一下Ctrl + C （Mac下是Command + C），系统会把所选的东西放到黏贴板，替换掉原有的东西。我需要一个工具能够存放多于一个东西的黏贴板，这样它就成为一个临时存储区域，可以方便地把东西放到这个区域中，随时复制到其他地方。Windows系统上我用的是ClipX，在Mac上我用的是ClipMenu。

以上这些工具能够极大地保证我的学习和工作效率，保证在电脑前所有的时间都用来做有意义的事情。这些工具是日常的一部分，我的工具箱在不断地调整，我也在不断地尝试比较新的工具。读者可以以这些为基础，建立自己的工具箱。这里还想分享一下选择工具的原则：

首先，不会为了选工具而选工具，选择的工具一定可以解决某个问题，这也就是为什么在分享工具的同时，也在分享这些工具解决的问题的原因；其次，选择能够发展自己能力的工具，比如Git，AutoHotkey等；最后，要不停地尝试和比较工具。

作者简介 | 滕振宇（Daniel）

目前国内唯一的一位认证Scrum教练（Certified Scrum Coach）。Scrum联盟Certified Scrum Coach以及Certified Scrum Professional评审委员会成员，敏捷全球之旅董事会成员。

■ 责任编辑：高松（gaosong@csdn.net）

C++模板元编程初探

■ 文 / 罗剑锋

C++是一种功能强大到几乎可称为“万能”的语言，支持许多种不同的编程范式，而由泛型编程衍生出的模板元编程（Template meta-programing，简称元编程）则无疑是其中最复杂、最强大和最具威力的一种，可算得上是C++的“终极”技巧。

元编程也有译称“超程序”或“超编程”，这个译法一定程度上反映了其本质——它是一种位于普通程序之上、超越普通程序的程序，是一种可以操纵、产生程序的程序。

模板元编程本质上是泛型编程的一个子集，从广义上来说，所有使用Template的泛型代码都可以称作是元程序，因为泛型代码不是真正可编译执行的代码，它们只是定义了代码的产生规则，是用来生成代码的“模板”。

然而模板元编程又不完全等同于泛型编程，它是一种“函数式编程”，并且已经被证明是图灵完备的，也就是说它具有足够的“计算”能力，可以“计算”任何东西。模板元编程的执行是在编译期，它把编译器变成了元程序的解释器，可以把C++的类型体系像面团一样捏来捏去，肆意打碎再任意组合起来，拥有近乎不可思议的“魔力”。

C++模板元编程已经诞生了十多年，但因为它过于抽象和艰深，许多程序员仍然对它感到陌生。本文将结合作者自身实践模板元编程的一些体会和理解，对模板元编程做一个初步的探讨，权作抛砖引玉之举。

模板元编程基本概念

虽然模板元编程使用的是标准的C++语言，遵循同样的语法规则，但它在编程思想、编程范式等很多方面都与普通的运行期C++程序有很大的不同。模板元编程产生的元程序是在编译期执行的程序，因此不能使用运行期的C++关键字（如if、else、for），可用的语法元素相当有限，最常用的是：

enum、static。用来定义编译期的整数常量；

typedef。最重要的元编程关键字，用于定义元数据；

template。模板元编程的“起点”，主要用于定义元函数；

“：：”。域运算符，用于解析类型作用域获取计算结果（元数据）。

下面简单介绍模板元编程领域中的五个基本的概念，熟悉它们有助于我们进一步理解模板元编程和元程序。当然元编程中远不止这些，还有Lambda元函数、序列、容器、迭代器、算法、视图等更高级的概念，但对于本文来说已经足够了。

元数据（Meta-Data）

元编程可操作的数据就称为“元数据”，也就是C++编译器在编译期可操作的数据，它是元编程的基础。元数据都是不可变的，不能够就地修改，最常见的元数据是整数和C++的类型。

对于整数大家都很熟悉，普通程序在运行期也可以很容易地处理。但在模板元编程中的元数据更多的是以类型（Type）的面目出现，这些元数据不是普通的运行期变量，而是如int、double、class（非模板类）这样的抽象数据类型——这是模板元编程与运行期编程的一个最根本的不同点，也是元编程的威力所在和令初学者感到最困惑的地方。

如果对元数据再进行细分归类，则元数据又可以分成值元数据（int、double等POD值

类型）、函数元数据（函数类型）、类元数据（class、struct等用户自定义类型）等。

使用typedef关键字可以任意定义（声明）元数据，很像运行期的变量定义语句。

元函数（meta-function）

元函数是模板元编程中用于操作处理元数据的一种“构件”，可以在编译期被“调用”，因为其功能和形式类似运行期的函数而得名，是元编程中最重要的构件。

元函数实际上表现为C++中的一个类或者模板类，它的通常形式是：

```
template<typename arg1, typename
arg2,...>
struct meta_function
{
      typedef some-define type;
};
```

编写元函数就像是编写一个普通的运行期函数，但形式上却是一个模板类：函数参数列表圆括号()变成了模板列表声明的尖括号<>，函数的形参变成了模板参数（即元数据）。因为元编程不能使用return关键字，所以元函数不能像普通函数那样返回计算结果，而是需要在元函数（模板类）内部用typedef定义一个名为type的类型（元数据）或者名为value的值作为返回 。还要注意的一个小区别是元函数最后要以分号（;）结束，这是因为元函数实质上是一个类，C++的语法要求类定义需要分号。

进一步类比普通函数有助于我们更好地理解元函数。

元函数同样也有形参、实参的概念，元函数的形参就是模板参数列表中的模板参数，实参就是元函数被调用时的真正类型（元数据），元函数的调用就是编译器对模板的实例化计算依赖的类型。元函数也可以没有返回值（即不定义内部类型type），也可以有重载，也可以有缺省参数，也可以分为无参、单参、多参等类别。但元函数没有普通函数参数传值、传引用的区别，也没有指针的概念，而且如果需要，元函数可以返回任意多个返回值。

请读者谨记一点，元函数就是一个形式上很像函数的一个模板类，它用于计算类型。

高阶元数据（high-order meta-data）

高阶元数据是一种特殊的类元数据，它内嵌有名一个为apply的元函数，形如：

```
struct high_order_meta_data
{
      template<typename T1, typename
T2, ...>
      struct apply
      {
            typedef T1 type;
      };
};
```

高阶元数据这个名词是笔者自行发明的，现有的元编程资料并没有使用这个词来称呼这种元编程构件，在Boost文档中使用的名称是“元函数类”。但笔者个人认为这个名词欠妥，因为在模板元编程中已经不存在C++的class概念了，所有的类型都作为元数据出现，使用“类”这个称呼容易造成概念上的混乱。

高阶元数据主要的功能是包装元函数，把它变成元数据，从而可以把元函数传递给其他的元函数进行调用。从功能来看，它的作用颇类似函数对象。

高阶元函数（high-order meta-function）

与高阶元数据对应，操作或者返回高阶元数据的元函数被称为高阶元函数，是一种更为复杂和强大的元函数。

元函数转发（meta-function forwarding）

这是模板元编程中一个经常用到的惯用法，相当于运行时的函数转发调用，但在模板元编程中则要使用public派生实现，模板参数传递给父类完成元函数的调用，这样子类会自动获得父类的::type定义，完成了元函数的返回。

理解了这些模板元编程基本概念后我们还可以深入考察其他既有的库，它们实际上已经或多或少地实践了模板元编程的理念。例如std::tr1::result_of，它能够推导出可调用类型的返回值，以result_of<T>::type的形式给出，因此它就是一个元函数。还有std::tr1::ref的unwrap_reference<T>，它能够解开boost::reference_wrapper的包装；boost::call_traits，它能够推导出最合适的调用参数类型。这样的例子还有很多，它们的类型“推导”实际上就是模板元编程中元函数对元数据（类型）的计算。

模板元编程工具

Boost是一个功能强大、构造精巧、跨平

台、开源并且完全免费的C++程序库，由C++标准委员会部分成员所设立的Boost社区开发并维护，有着“C++‘准’标准库”的美誉。

在C++0x为模板元编程提供正式的语言级别支持之前，Boost几乎是以一己之力创立并发展了模板元编程，以库的方式提供了许多可用于模板元编程的工具，构建了一个完整而精致的元编程大厦。

Boost提供三个用于元编程的库，它们是：

- type_traits：提供一组特征（trait）类，即单参元函数，可以在编译期确定类型（元数据）是否具有某些特征，例如是否是原生数组，是否是整数。此外它还提供了判断类型之间的关系和操作类型的元函数，可以检查两个类型是否是同一个类型，或者为类型增添或移除const、volatile等修饰词。type_traits以库的方式实现了人们原本以为必须扩展C++语言才能实现的强大功能，是泛型编程和模板元编程所必须的基础设施。
- function_types：它专门处理函数元数据（函数类型），可以对函数类型中包含的元数据进行任意的分解和合成，较type_traits中的function_traits而言它处理的更加精细、准确和方便（或许库的名字改叫做function_type_traits会更为恰当）。
- mpl：它是模板元编程最主要和最重要的工具，定义了元编程中的各种概念，提供完整且强大的元编程武器，包括断言、各种控制结构、仿STL风格的容器和算法（当然它们容纳、处理的都是元数据（即类型），而不是真实的数据），以及编译期的lambda表达式。

Boost中还有另外的两个辅助工具用于元编程的正确性验证：

- static_assert：编译期断言，相对于运行期的assert宏又称为“静态断言”，它提供了在编译时的诊断宏，是模板元编程中最易用、也是最常用的诊断工具。
- concept check：类似C++0x草案中被否决的概念检查特性，提供了大量的标准概念检查元函数，能够在编译诊断时给出更可读的信息。

由于C++98标准的语言限制（没有可变模板参数列表），模板元编程还经常与预处理元编程库Boost.Preprocessor结合在一起使用，以获得更多的灵活性和更大的方便。

模板元编程“伪关键字”

模板元编程是一种全新的C++编程范式，但它却仍然使用原有的C++语法，通篇的typedef、template关键字使元程序看起来有如“天书”，对于初学者来说通常很难在短时间内适应这种转变。笔者在实际开发工作中使用宏定义了一些模板元编程的“伪关键字”，一定程度上能够使模板元程序更加清晰易懂，更能够显示其元编程的本意，也确实收到了较好的效果。

这些自定义“伪关键字”均以mp_开头，如下：

```
// mp_utils.hpp
#define mp_arglist              template
#define mp_arg                  typename
#define mp_function     struct
#define mp_data         typedef

#define mp_return(T)    mp_data T type
#define mp_exec(Func)   Func::type
#define mp_eval(Func)   Func::value
```

使用这些“伪关键字”，之前的元数据和元函数可以改写成如下的形式：

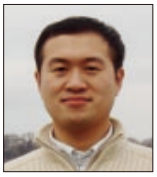
```
mp_data int meta_data1;
mp_arglist<mp_arg T1, mp_arg T2>
mp_function select1st
{
      mp_return(T1);
};
```

不过读者需要注意的是由于宏自身的限制，后三个“伪关键字”的作用有限，它们只能处理简单的参数，如果是带有逗号（,）的模板类就会失效。

结论

模板元编程是一种全新的编程体验。理解了它的原理，能进一步加深对于C++语言的认识，进而更加充分地利用C++的类型系统和发掘编译器的能力，最终提高真正程序的质量和执行效率。

作者简介 | 罗剑锋



2003年毕业于北京理工大学，获计算机专业硕士学位。任某部委下属软件公司项目经理，主要研究方向为C/C++、设计模式、密码学、数据库、嵌入式系统开发。业余爱好是阅读、欣赏音乐和旅游。

■ 责任编辑：高松（gaosong@csdn.net）

强大的开源全文检索引擎 —— Sphinx

■ 文 / 乔楚

随着互联网的发展，Web 2.0带来了信息的井喷，我们可以接触和处理的数据越来越丰富，规模也越来越庞大。不管是互联网网站还是企业自身，都需要对大量的数据进行管理和分析，并且用户对信息的组织、查询、可寻性、及时性的要求也越来越高，这一切都离不开全文检索引擎。

后来居上的Sphinx

以往提到开源全文检索引擎，大家都会想到基于Java的Lucene及其衍生品。今天，我们要说的是Sphinx，它是目前仅次于Lucene的基于C++的开源全文检索引擎。可以说，虽然是全文检索引擎的小字辈，却获得了后来居上的份额。

Sphinx是一个在GPLv2下分发的全文检索引擎，为了高性能的全文检索而诞生。来自俄罗斯的Andrew Aksyonoff从2001年开始开发Sphinx，2006年建立站点提供公开使用，到今天Sphinx已经走过十个年头。

从2006年开始，我们基于Sphinx开发了专门针对中文环境使用的版本 —— CoreSeek，同样在GPLv2下分发，并为国内用户提供完善的本地化技术支持服务。

Sphinx的主要特性

作为一款高性能的全文检索引擎，Sphinx具有以下主要特性。

- 基于C++开发，相对于基于Java的Lucene，绝对称得上短小精干，同时支持在所有主流的Linux、BSD、MacOS、Solaris、Windows等操作系统上运行。
- 内建对MySQL、PostgreSQL、ODBC的支持，还可以通过XML数据源获取XML格式的数据；CoreSeek还为Sphinx开发了Python数据源，充分利用Python无与伦比的数据操作能力，几乎支持所有你可以见到的数据库系统，并可以支持目前炒得火热的NoSQL类系统，以及任何其他可以被Python操作的数据。
- 索引数据高效快速。在主流的台式电脑上可以达到每秒索引10~15MB的文本的速度，在更高级别的服务器上可以达到超过每秒索引60MB文本的高速度；索引100万条数据可以在5分钟内完成，索引1000万条数据可以在1小时内完成，而只包含最新10万条数据的增量索引，重建一次只需数十秒。
- 查询性能卓越。在内存2GB的双核台式电脑上，针对1.2GB大小的100万文档，可以达到每秒500次的查询速度。
- 支持分布式搜索，可以将索引分布到多台机器，提高搜索系统的负载能力和高可用性。
- 具有丰富的高级全文检索手段，包括多种关键字匹配模式，更好的评分计算方式，丰富的结果排序模式，以及结果聚类分组。
- 提供丰富的查询接口，易于被应用程序调用，已经提供的接口包括：PHP、Perl、Python、Ruby、Java、C/C++、C#等；同时，还支持使用SQL语法进行查询。
- 提供对互联网黄金搭档PHP-MySQL的完美结合，提供了PHP扩展和PHP代码的查询接口，可以作为MySQL存储引擎使用，自身也可

以作为一个专用于全文检索的MySQL服务器使用。

Sphinx可以处理SQL数据库的数据、NoSQL存储系统的数据，以及其他可供操作的文件或者数据，从而使得其应用范围非常广泛。

其最主要的应用场合，是为数据库的数据提供全文检索服务，从而降低数据库的负载，提升整个系统的承载能力，可用于站内检索、论坛搜索、垂直搜索等。

因为具有丰富的高级全文检索手段，Sphinx还被用于文档检索、文献检索、情报检索、舆情检索、日志分析、局域网检索、企业内部检索等各种场合。

最新测试版本的Sphinx还提供了实时索引，可以用于实时搜索的场合。

和其他知名的开源全文检索引擎，Sphinx有其优点，也有其不足。

相对于Lucene而言，Sphinx的优势在于：

- 基于C++，高效快速，运行稳定；
- 建立索引速度Lucene望尘莫及，比Lucene快5~10倍；搜索速度快于Lucene；
- 对数据库的支持全面完善，还可以直接作为MySQL的存储引擎；
- 无需二次开发，即可直接使用；
- 拥有丰富的全文查询语法。

Sphinx的不足在于：

- 索引的数据都需要有一个唯一整数编号；
- 无法像Lucene快速嵌入其他应用内部，需要作为服务运行，通过接口调用；
- 分布式搜索没有Lucene完善和强大，需要人工进行数据的切分；
- 索引数据中的文档内容不能及时更新，且不能保留原始文本内容；最新测试版本支持，但尚处于测试阶段。

另一个知名的基于C++的开源全文检索引擎是Xapian，具有20多年的历史，类似于Lucene可以嵌入到其他应用使用。相对Xapian而言，Sphinx更具有优势：

- 开发团队活跃，社区支持活跃，文档详细，案例丰富；
- 功能更全面，可靠性更好；
- 分布式搜索更灵活；

获得广泛的使用

目前，国内外已有不少网站使用Sphinx作为全文检索引擎，为用户提供在线搜索服务。

在国外，已知最大的Sphinx搜索集群由BoardReader建立，数据记录量超过50亿条，文本容量超过6TB。

最繁忙的Sphinx搜索集群由全美排名前十的站点Craigslist建立，每天提供超过5千万次查询。

其他使用的还包括DailyMotion、NetLog、Mozilla扩展站点、MySQL官方Eventum缺陷跟踪系统等等众多站点和应用。

在国内，使用Sphinx的网站也越来越多，目前已知的部分网站包括：

- ChinaUnix —— Linux/Unix应用与开发者社区；
- 博客大巴 —— Blog托管服务商；
- 易查 —— 手机搜索引擎；
- 金山逍遥 —— 金山游戏官方网站；
- dospy —— 塞班智能手机论坛；
- UCWEB —— 手机浏览器；
- 米胖网 —— 旅游网站；
- PPLive —— 视频网站；
- 车市 —— 汽车网站。

同时，国内主流的通用社区论坛软件系统Discuz!、PHPWind，国内领先的网站管理系统PHPCMS等，也纷纷选择Sphinx为其提供全文检索功能，在最新官方版本中均集成了对Sphinx搜索服务调用的支持。

随着Sphinx社区的不断发展和众多网站的采用，Sphinx将会获得更大的发展。值得开发者加以关注。

作者简介 | 乔楚

长期活跃在ChinaUnix等网络技术社区，狂热的开源爱好者和传播者；在ChinaUnix担任Web服务器、PHP、Web开发、互联网行业四个版面的资深技术版主。

■ 责任编辑：高松（gaosong@csdn.net）

Flip Phone

这款引人入胜的概念手机运行在Android系统上，由三块可自由翻转的屏幕组成，可以通过翻转组合成各种不同的形状。每一面都是曲面强化玻璃显示屏，三块屏幕既可以组成大屏幕来浏览网页，也可以分屏显示系统的不同部分。其中一块屏幕的背面还有一个完整的QWERTY键盘，并内置摄像头。

GEEK产品

Koko Muo

KoKo Muo这款手表应用了极度简化的设计风格。它的表盘里面没有数字刻度，只是用两个区域分别表示小时和分钟，并用对比度强烈的黑白两色来表示传统手表指针的移动。

JoyFactory Zip

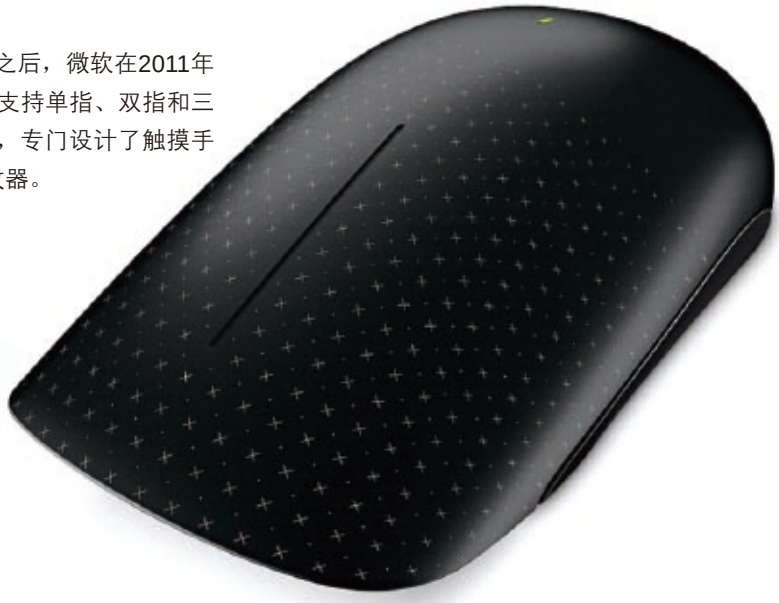
JoyFactory推出的一款用于多种设备充电的装置，通过不同类型的转接口，支持不同的设备同时充电。

Chobi Cam One

喜欢小巧东西的你有福了！日本的JTT最近发布了可能是世界上最小的单反相机，重12g，可拍摄1600×1200的照片和1640×480的视频。它采用miniSD卡存储，USB充电。最绝的是，这么小个儿的相机居然可以换长焦及广角镜头。

Microsoft Touch Mouse

继去年的带滚轮触摸功能的Arc Touch Mouse之后，微软在2011年CES大展上又推出了全触摸设计的Touch Mouse。支持单指、双指和三指触摸手势操作，同时针对Windows 7的快捷操作，专门设计了触摸手势功能。采用蓝色LED光学引擎，提供USB无线接收器。



LinkSys E4200

印象中，无线路由器都是白色的扁盒子，上面带几个插口，看起来不那么有趣。Cisco旗下的LinkSys最近推出的E4200，或许能够彻底颠覆人们的这种印象，外媒对其工业设计的评论甚至用到了“性感”这个词。

Camera Futura

Arteface Group展示了它心目中未来数码相机的样子：WVIL Camera Futura。配备3100万像素全片幅CMOS感光组件，支持无线、蓝牙、GPS。相机镜头更能够与屏幕分体成为独立部分，可以利用无线方式进行变焦、拍摄。相机内置的软件采用完整的智能手机操作系统，能够上网、下载软件、上传图片等。

新产品&新工具

Google API和开发者产品元素表

Google开放了海量的API，为了方便开发者调用，Google制作了一个“Google API和开发者产品元素表”，里面列出了至2011年1月为止所有API和跟开发者有关的产品，涵盖移动、搜索、Gadgets、数据API、社会化、广告、地图、工具、Chrome等等方面，点击每个“元素”即可了解更多API信息。

数据质量分析DataCleaner 2.0 发布

DataCleaner是一个数据质量分析、比较、验证和监督的软件。DataCleaner包括一个独立的图形用户界面分析、比较和验证，并进行监测Web应用。

网上购物系统PrestaShop

PrestaShop是一款针对Web2.0设计的全功能、跨平台的购物车套件，能够部署在支持PHP 5的服务器端。PrestaShop具有良好的自定义性，安装方便，且体积非常轻巧，整个程序只有大约6M。其后台以电子商务模式设计，功能强大。主要特性包括：支持不限制的支付方式、RSS、多种搜索方式、商品描述和用户评论、商品图片的缩放、商品预约、短消息提醒功能、多语言支持等等。

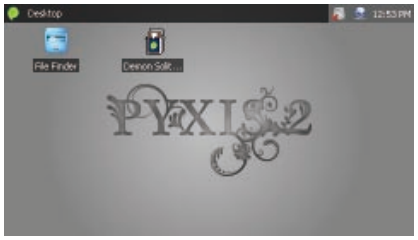
Guile发布2.0版

Guile是Scheme语言解释器/虚拟机项目，最新的2.0版本新特性包括：显著改进性能，支持ECMAScript和Emacs Lisp，新的调试工具REPL，支持健康的宏（hygienic macros），支持Unicode，部分兼容R6RS标准（最新的Scheme语言标准），新的动态外部函数接口，换用Boehm-Demers-Weiser垃圾收集器，新的模块等。

MeshLab 1.3.0 发布

MeshLab是一个网格处理系统，为用户辅助编辑、清洗、筛选和渲染大型结构的三维三角网格（典型三维扫描网格），它可以帮助处理在3D扫描捕捉时出现的无特定结构的模型。该系统依靠了网格处理任务GPL的心向量图库。

MeshLab 1.3.0引入了很多很好的功能，点云管理被大幅提高，增加导入Photosynth Datasets的功能，以及统一的光栅数据。

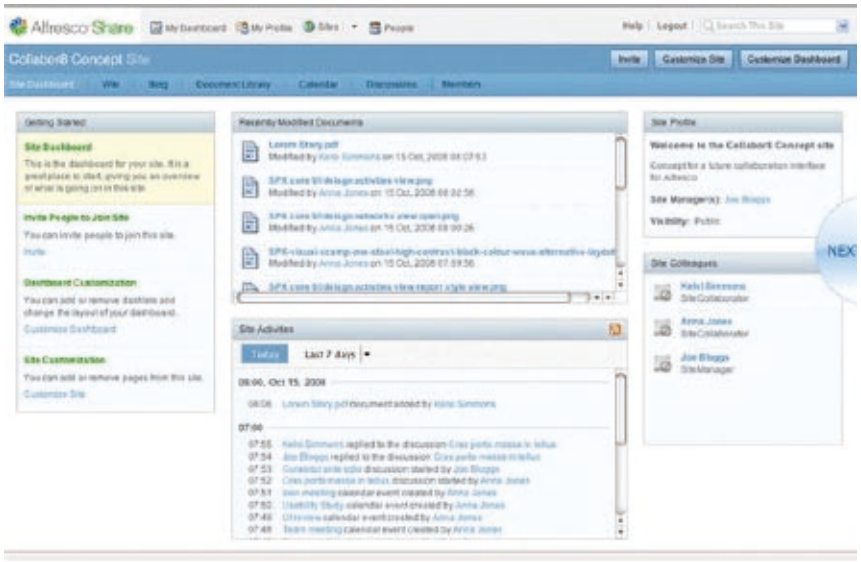


Pyxis 2“操作环境”发布

Pyxis 2是一个在.NET Micro Framework下用C#编写的一个操作环境。该项目的目标是让开发人员开发强劲的NETMF程序，同时提供用户一个共同的环境以启动应用程序 。

轻量级jQuery插件FancyBox

FancyBox是一款基于jQuery开发的类Lightbox插件。支持对放大的图片添加阴影效果，对于一组相关的图片添加导航操作按钮，该lightbox除了能够展示图片之外，还可以展示iframed内容， 通过CSS自定义外观。



企业文档管理系统 Alfresco

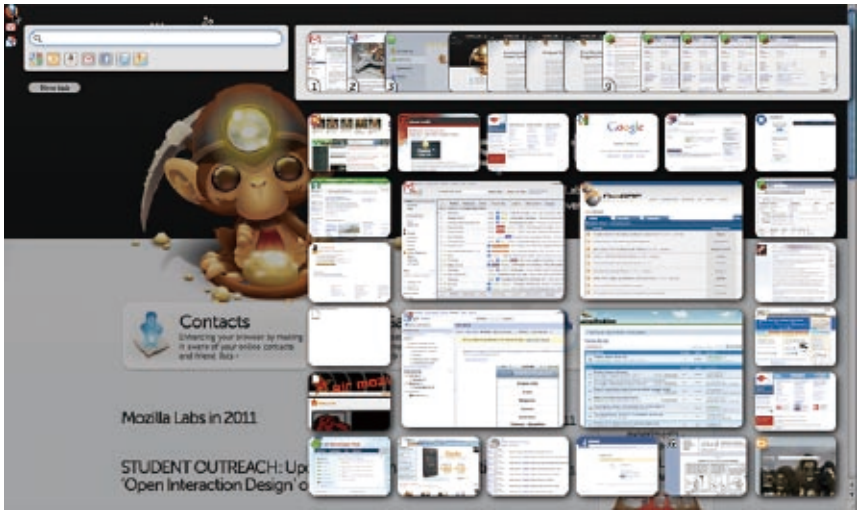
Alfresco 提供了开源的企业内容管理系统（ECM），功能包括：文档管理、协作、记录管理、知识库管理、Web内容管理等功能。其页面采用FreeMarker 开发。

VMware发布下一代邮件和协同平台——Zimbra 7

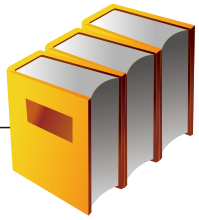
VMware公司日前正式发布了下一代邮件和协同平台的最新版本——VMware Zimbra 7，为IT和终端用户提供了新的数据分享、扩展的日历和搜索功能以及升级的管理功能。其新功能包括：管理员支持、增强的移动体验、电子邮件和短信提醒等。

Home Dash

Mozilla Labs 中的 Prospector 项目组又带来了一个新发明：Home Dash。Home Dash 的目的是带来称之为“浏览为主的浏览器”体验，使用Firefox 4中的Jetpack技术（无须重启立即生效）实现。



新书上架



门后的秘密：卓越管理的故事

作者：Johanna Rothmann、Esther Derby
译者：于梦瑄
出版社：人民邮电出版社

这是被软件管理大师Jerry Weinberg赞誉为“没有辜负我多年来的期望”的小册子。本书的两位作者都是IT界知名的管理顾问。在书中，她们将为大家揭示卓越管理的秘密。书中以一名卓越的管理者每周的工作为主线，详细介绍他如何与下属共同工作，克服重重困难，最终形成一个高效多产的团队。

作者没有大书特书管理的规则，也没有简单罗列管理工作的要点清单，而是着重还原一个个真实生动的管理活动场景（一对一面谈、部门例会等），展示卓越的管理者是如何安排日常工作、迎接挑战、应对危机的。在此基础上，作者也用一定篇幅总结出许多实用的经验和技巧。

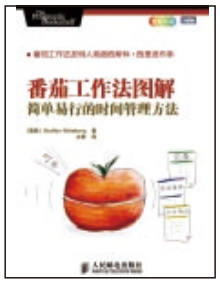
Android系统级深入开发——移植与调试

作者：韩超 梁泉
出版社：电子工业出版社

本书以移植和调试为重点，全面了介绍Android系统级开发的作品。

从开发者的角度出发，全书特点主要包括以下几个方面：使用的代码以Android的开源工程为主，硬件也是比较常见的设备，保证读者可以很容易地获得开发环境；根据Android系统自身的固有特点，从Linux内核中的驱动和硬件抽象层两个着手点作为切入点。对于Android中规模和难度各不相同的子系统，抓住它们和硬件相关的共同点，采取同样的格式和思路进行介绍，体现了“从特殊到一般”的理念；简要介绍各个子系统的框架，并列出代码路径，对移植部分的主要调用部分加强提示，让读者更全面地把握系统。

番茄工作法图解



作者：Staffan Noteberg
译者：大胖
出版社：人民邮电出版社

番茄工作法是弗朗西斯科·西里洛在1992年发明的。在大学生活的头几年，他曾一度苦于学习效率低下。几次三番地挣扎后，他终于找到了利用身边最简单的工具（一枚厨房定时器，形状像“西红柿”），设定固定的时间，全力地工作和学习的方法，并命名为番茄工作法。简而言之，番茄工作法是一套简单的工具和流程，是“25分钟里只做一件事”强制方法，是可以提升个人和所在团队生产力的有效武器。

对于每天处在进度紧张巨大压力下的程序员来说，协助个人和团队应用番茄工作法，改进运作流程，优化时间安排，可以把工作变得张弛有度、提高工作效率，也就离成功更近了。本书作者、译者都是番茄工作法的成功实践者，有他们的现身说法，番茄工作法不会让人失望。

竹林蹊径：深入浅出（Windows驱动开发）

编著：张佩 马勇 董鉴源
出版社：电子工业出版社

本书主要包含以下几个方面的内容：WDF框架、驱动测试、音视频编程、驱动安装。这其中，作为重头戏的是WDF框架相关章节。WDF是目前和将来驱动开发的大势所趋。如果初学者因为资料的关系，而固执在WDM架构上的话，可能正在错过一旁正如日中天的WDF。

本书详细介绍了使用WDF框架进行USB和1394编程的内容。为配合USB一章的写作与学习，作者专门请朋友精心设计了一款USB驱动学习开发板。驱动测试方面包括两章内容，分别对WDF驱动测试和内核调试利器WinDBG的调试命令详细介绍。

本书最后三章，介绍驱动安装有关的知识，包括驱动安装的原理及系统模块、INF安装文件的技术细节、如何编写驱动安装软件。

项目百态：深入理解软件项目行为模式

作者：Tom Demarco、Peter Hruschka、Tim Lister、Suzanne Robertson、James Robertson、Steve McMenamin

译者：金明

出版社：人民邮电出版社

本书是第19届Jolt大奖获奖作品。作者清楚地讲述了项目因何失败，有何补救措施，但并不以专家/过来人自居，而是以同行的亲切口吻和令人乐于接受的方式提出了切实可行的建议。

本书介绍了软件项目行为的86个模式，基本上概括了软件项目生命周期的全部，揭示了软件项目最常遇到的困境，反省了行业内种种不良习惯和做法。六位作者均来自开发咨询的团队Atlantic Systems Guild，长期以来为众多软件公司提供咨询服务。他们浓缩了成百上千个项目管理的案例，通过本书中一个个模式展现出来。每个模式都以生动形象的插图开始，另外还加上一些趣闻和真实事件。

SNS网站建构

作者：Gavin Bell

出版社：机械工业出版社

SNS是Web 2.0的典型应用，Facebook帝国的日益强盛让人见识到了SNS的影响力和其中蕴含的巨大机遇。然而开发SNS也是需要精心细致地规划的，本书就条分缕析地介绍了SNS网站建构的方方面面。

本书的一个核心表述是“万变不离其宗”。作者Gavin Bell根据自己多年来在不同领域的跨界经验，极为杂糅地涵盖了从社区、软件开发、视觉设计、社会化媒体、管理、社交网络伦理、社会网络模式、网站组织、API设计等各个领域，并且总结出了有效的实践模式。无论何种商业模式的SNS都可以从中有所借鉴。通过这本书的学习，SNS开发者学习到网站建设初期准备、克服中期发展瓶颈、保持后期创新力等宝贵教益。

Linux内核设计与实现（英文版·第3版）

作者：Robert Love

出版社：机械工业出版社

本书作者Robert love是一位传奇性的人物。他还是大四学生的时候，设计了Linux的抢占式内核，这是2.4到2.6版内核的最关键进步之一。

整本书很薄，但是内容颇为丰富，对2.6增加的若干改进有着非常好的描述。通读一遍就能对内核有整体的把握，不至于陷入细节的纠缠中，的确适于进阶者阅读。本书基于Linux 2.6内核介绍了Linux内核的设计与实现，涵盖了从核心内核系统的应用到内核设计与实现等各方面内容，主要包括：进程管理、调度、时间管理和定时器、系统调用接口、内存寻址、内存管理、页缓存、VFS、内核同步、可移植性、调试技术等。此外，本书还讨论了Linux 2.6颇具特色的内容，包括CFS调度程序、抢占式内核、块I/O层以及I/O调度程序。

全球排行榜

	Amazon
1	Head First Java, 2nd Edition
2	The Facebook Effect
3	The Web Designer's Idea Book
4	Beginning iPhone 4 Development: Exploring the iOS SDK
5	Head First HTML with CSS & XHTML
6	Programming in Objective-C 2.0
7	Head First Design Patterns
8	Hello, Android: Introducing Google's Mobile Development Platform
9	PHP and MySQL Web Development
10	Introduction to Algorithms, Third Edition

	天珑书局（中国台湾）
1	Google Android 2.X应用程序开发实战
2	Google Android SDK开发范例大全 2
3	深入浅出Android游戏程序开发范例大全
4	深入浅出Android系统原理及开发要点
5	HTML5 & API网页程序设计
6	最严选PHP案例模块开发讲座
7	iPhone创意程序设计家, 2/e
8	Android应用开发揭秘
9	鸟哥的Linux私房菜——基础学习篇, 3/e
10	王者归来Java Web整合开发——JSP + Servlet + Struts + Hibernate + Spring

	第二书店
1	深入理解计算机系统
2	Android应用开发揭秘
3	设计原本：计算机科学巨匠Frederick P. Brooks的思考
4	竹林蹊径：深入浅出Windows驱动开发
5	深入理解计算机系统
6	高效程序员的45个习惯：敏捷开发修炼之道
7	程序员的思维修炼:开发认知潜能的九堂课
8	精通Qt4编程
9	Linux系统移植
10	HTML5高级程序设计

互联网商旅预定服务的未来兴衰

王煜全，Frost & Sullivan中国区总裁。

记得前几年坐地铁时，经常会在地铁的视频中看到一则广告——徒弟对师傅说：“师傅，我们去旅游吧，直接途牛，旅途的途，牛人的牛”。这段时间，正是携程、e龙等互联网酒店机票预订网站繁荣发展的时候。在出行时，我们很自然地会到这些网站上去查询并订购相关的酒店和机票，这也已经成为多数人的首选。

携程这类网站的出现顺应了当时用户需要，也是IT发展水平下顺应而生。随着经济发展，到各地出差办事、探亲访友，或者休闲旅游的人逐渐增多起来，这时人们发现要想顺利查询到酒店以及机票信息并且能够方便地订到相关的产品，流程变得繁琐复杂。能不能有一种整合的力量来给用户解决这些难题？于是就诞生了携程这种类型的中介服务型网站。而携程类网站之所以快速发展，除了用户的需求之外，还有一个重要的因素就是当时很多酒店在花大力气去建设硬件设施，去扩展区域占据有力位置，因此还没有精力去考虑自己建设相关的IT系统。

随着携程类网站的发展，其话语权一再增加，使得很多酒店发现通过携程类网站来接触客户会变得有阻碍：客户前期不直接接触酒店；网站依靠对客户的掌控对酒店提出很多要求。同时，一些酒店，特别是连锁酒店也完成了初期的硬件设施建设和区域发展，意识到了IT软件服务的主要性，自己建设了相应的预定系统，比如汉庭连锁、如家连锁、7天酒店、速8酒店等。

因此，我们将预见到未来的酒店预定服务会出现如下的变化：各酒店会直接面向用户提供IT手段的预定服务，这已经是事实；各酒店未来将开放自己预定系统的接口，提供酒店的房间预定信息，包括价格、时间以及服务内容等；各种互联网以及手机上的应用可以随时调用这些酒店的开放信息来构成其服务用户的内容，也就是各种应用将成为酒店的渠道。

那么在这种情况下，我们有什么机会呢？

- 最简单，也是最没有持续性的机会：帮着各连锁酒店开发基于手机客户端的软件，这属于技术外包的领域，基本没有进一步发展的机会。
- 与各酒店进行数据对接，然后将这些酒店信息数据进行标准化处理，规范成API，将包含酒店各种数据的API开放给各种应用，通过各种应用的调用和内嵌，使得各种应用成为接触用户的渠道。当然，这个机会实现起来难度很大。因为酒店的信息数据整合就是一个繁重的工作。值得庆幸的是，目前已经有一些公司在做这方面的整合了。
- 利用酒店数据的API进行针对性应用开发。对于一些个人开发者或者小团队开发者来说，利用现成的API接口数据进行针对性开发是个不错的选择。例如在微博，开发一个博主发起的酒店预定计划，一个酷爱旅游的达人设计一条好的旅游线路后，在自己微博上发布，召集粉丝一起参加。这样其微博处就会出现一个酒店预定的图标，参加活动的人能够通过这个图标接入到酒店信息，预定房间。当然这里面包含了团购的味道，那么使用者也可以反过来和酒店进行协商，拿到更好的价格和服务。同时，要想实现服务，支付的功能也是必需的，这当然也可以调用API来完成。

上述内容其实就是顺应移动互联网开放发展的原则，在基础的开放API之上完成业务设计，这不但是未来商旅预定的发展，也是整个移动互联网的发展，机会无限！

高巍，安卓爱普公司创始人。原搜狐媒体产品中心经理。关注移动互联网、互联网产品管理。
微博：
httpt.sina.com.cn/inetpm

程序员如何应用“刻意练习”

随着Malcolm Gladwell的《异类》、Geoff Colvin的《哪来的天才》等畅销书的流行，“成功的一万个小时”概念逐渐深入人心。通俗的说法就是，想要在任何领域取得卓越成就，需要至少一万个小时的“刻意练习”。

CSDN上最近一篇颇受关注的文章《软件天才都是训练出来的》，也谈到了这个话题：软件天才，或者至少说软件人才，是可以通过训练培养出来的。但具体如何训练，文中只是一带而过、语焉不详。有意思的是，国外的技术问答社区StackOverflow，有个帖子讨论得很火，说的正是“How does a programmer employ deliberate practice?”。

程序员进行“刻意练习”，最早是在《Software Craftmanship》一书中正式提到，新出的《程序员应该知道的97件事》也有一小节提及。但最系统、详尽讨论的是《Apprenticeship Patterns: Guidance for the Aspiring Software Craftsman》，可以称得上是程序员“刻意练习”的一本行动教科书。

你需要重新思考自己的价值观和幸福观

真正的“刻意练习”，尽管其结果是超越自我的愉悦，但过程却是非常痛苦的。在学习中，需要不断人为地设置障碍，再调动各种资源去攻克它，一次又一次有意识地主动离开自我的“舒适区”。惭愧地说，大部分人包括我自己，很难坚持这一过程，正是“刻意练习”的这种难度和挑战把不同的人明显地区别开来。10年前，我第一次读到了侯捷老师的文章《MFC四大天王》，后来还有荣耀的《C++程序设计之四书五经》，这些都是编程学习的路线图，列出了从基础到提高的一系列必读书籍。当时也曾雄心万丈要啃下来，但学习到《C++标准程序库》时却退下阵来，以至于多年以后遇到该书译者孟岩老师，很内疚地说，你翻译了一本很好的书，但我却没有读完的毅力。

所以，希望投入到“刻意练习”的软件学徒，需要一个较为崇高的自我预期，需要持之以恒的意志力。一般人是差不多了，别再难为自己了。而有志“刻意练习”的软件学徒，是真想要达到一个高的境界，对自我有期许、有要求，故意和自己较真儿，如Marten Gustafson所说“本质就是这种对自身的关注和提高自身技能的要求。”

程序员如何应用“刻意练习”

“私相授受”的师傅带学徒，是比学校教育更有效的学习方式。“刻意练习”的精髓是要持续地做自己做不好的事，精确地在“学习区”内进行，要求有高度的针对性。在很多情况下，这需要一个好的师傅或者教练，毕竟从旁观者的角度更能发现问题。“刻意练习”必须因材施教，小班学习，甚至是一对一的传授。

如果自己身边没有好的师傅怎么办？软件工程专家林锐在《大学十年》总结的第一条经验，就是要去主动创造环境。其中提到周鸿祎对他的帮助，“于是我向只有一面之缘尚在北大方正工作的周鸿祎求助。当我小心翼翼地展示约10万行C++代码的软件时，他竟在十几分钟内就指出多处重大的设计错误。…… 周鸿祎放心不下，觉得我‘恶病需用猛药治’，于是意犹未尽地把我捉到北大方正插在他管辖的部门，让我学习怎样做事情。从北大方正‘劳改’了两个月回来，我心服口服地承认失败了。我把察觉到的数十个毛病列出来，日后一个一个克服掉。”

充分利用网络和开源的力量

程序员进行“刻意练习”，与其他领域相比有一个天然优势，就是可以充分利用网络和开源。除了《Apprenticeship Patterns: Guidance for the Aspiring Software Craftsman》提到的方法， Hacker News技术社区的讨论还建议了一些网络资源，如Coding Dojo、Code Kata，Ruby社区的Ruby Quiz邮件列表等。

优秀的开源代码也是很好的学习对象。StackOverflow问答社区建议，可以借鉴富兰克林学习写作的方法，分析一段优秀的开源代码，梳理逻辑、做好笔记，然后尝试自己重新实现，再与源代码进行比对。这个过程可以循环递进地进行。

Robbin: Unix程序员。AppleFans。Geek。急需女朋友。命令行狂热者。

Hank: Windows程序员.NET高手。已婚。保守派。

Ada: 架构师。第三性别，喜欢挖苦男人。热爱无所不能的emacs。

作者介绍：西乔
项目经理。06年起携创业团队从事Web技术外包开发及产品咨询顾问。

如果你有什么好玩的关于程序员的故事、对话、代码，愿意通过漫画的形式分享，请给西乔发邮件：arthur369@gmail.com。

幽　默

某程序员签名档：每当加班写Code的时候，就遥想当年诸葛亮造10万支箭的Project，第一天不干，第二天不干，第三天Deadline快到的时候，拉鲁肃通宵！

一程序员哭诉：我现在生活中唯一能接触到mm的地方，就是Android中编译当前模块时敲mm。

老板：才写了这么点啊？
程序员：如果再给我点时间，我会写得更短些！
老板：你可以回家休息了！
程序员：……

上联：为API生，为框架死，为Debug奋斗一辈子！
下联：吃符号的亏，上大小写的当，最后死在需求上！
横批：杯具程序员！

看到代码里面有一处qulonglong，想是谁这么高调把名字留在代码（不是注释）里面，而且名字还这么可爱，瞿龙龙？
哦，原来是Qt的无符号长长整型。

程序员去食堂吃饭，张口就说：师傅你们这里支持菜票吗？
师傅：我们这里兼容钞票和菜票。

一句话幽默

中午吃饭，又看到那家烧烤店了，叫“明月三千里”，旁边写着拼音缩写，华丽丽的“MYSQL”。

* { float:cloud; } = 神马都是浮云。

目前运算速度最快的计算机，跑完一个死循环只需要6秒。

每个编程问题都可以仅仅用一个抽象层（或间接地）来解决。

一个合格的程序员是不会写出诸如“摧毁地球”这样的程序的，他们会写一个函数叫“摧毁行星”而把地球当一个参数传进去。

程序员也是很Hip-Hop的职业，SVN更新好了，就大叫一声“Yo，check it out！”